UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Empirical characterization and modeling of power consumption and energy aware scheduling in data centers

Tesis de Maestría en Informática

Jonathan Muraña

Programa de Posgrado en Informática - PEDECIBA
Universidad de la República

Montevideo – Uruguay
Octubre de 2019

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Empirical characterization and modeling of power consumption and energy aware scheduling in data centers

Tesis de Maestría en Informática

Jonathan Muraña

Tesis de Maestría presentada al Programa de Posgrado en Informática - PEDECIBA , de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en .

Director:
Ph.D. Prof. Sergio Nesmachnow

Montevideo – Uruguay
Octubre de 2019

Muraña, Jonathan

Empirical characterization and modeling of power consumption and energy aware scheduling in data centers / Jonathan Muraña. - Montevideo: Universidad de la República,

XI, 111 p.: il.; 29, 7cm.

Director:

Sergio Nesmachnow

Tesis de Maestría – Universidad de la República, Programa en Informática - PEDECIBA , 2019.

Referencias bibliográficas: p. 102 – 111.

1. computación verde , 2. eficiencia energética , 3. multinúcleos, 4. modelos de energía, 5. simulación de computación en la nube. I. Nesmachnow, Sergio, . II. Universidad de la República, Programa de Posgrado en Informática - PEDECIBA . III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Prof. Pablo Rodríguez Bocca

Prof. Bernabé Dorronsoro

Prof. Esteban Mocskos

Montevideo – Uruguay
Octubre de 2019

ABSTRACT

Energy-efficient management is key in modern data centers in order to reduce operational cost and environmental contamination. Energy management and renewable energy utilization are strategies to optimize energy consumption in high-performance computing. In any case, understanding the power consumption behavior of physical servers in datacenter is fundamental to implement energy-aware policies effectively. These policies should deal with possible performance degradation of applications to ensure quality of service.

This thesis presents an empirical evaluation of power consumption for scientific computing applications in multicore systems. Three types of applications are studied, in single and combined executions on Intel and AMD servers, for evaluating the overall power consumption of each application. The main results indicate that power consumption behavior has a strong dependency with the type of application. Additional performance analysis shows that the best load of the server regarding energy efficiency depends on the type of the applications, with efficiency decreasing in heavily loaded situations. These results allow formulating models to characterize applications according to power consumption, efficiency, and resource sharing, which provide useful information for resource management and scheduling policies. Several scheduling strategies are evaluated using the proposed energy model over realistic scientific computing workloads. Results confirm that strategies that maximize host utilization provide the best energy efficiency.

Keywords:
green computing, energy efficiency, multicores, energy model, cloud simulation.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Data centers are key infrastructures for developing and executing industrial and scientific applications. In the last decade, data centers have become highly popular for providing storage, computing power, hosting, middleware software, and other information technology services, available to researchers with ubiquitous access (Buyya et al., 2013). However, energy efficiency of data centers has become one of the main concerns in recent years, having a significant impact on monetary cost, environment, and guarantees for service-level agreements (SLA) (Dayarathna et al., 2016).

The main sources of power consumption in data centers are the computational resources and the cooling system (Nesmachnow et al., 2015). Regarding power consumption of the computational resources, several techniques for hardware and software optimization can be applied to improve energy efficiency. For example, software characterization techniques (Anghel et al., 2016), which are applied to determine features that are useful to analyze the software behavior. This behavior analysis is an input to analyze and improve power consumption (Brandolese et al., 2011).

Energy consumption models are useful synthetic tools for studying and analyzing diverse issues related to energy efficiency of computing infrastructures, e.g., by predicting the power consumption of its components for a given workload. Information resulting from the analysis is often used by decision-makers to take technical, monetary, or environmental decisions regarding the computing infrastructure. Energy models are based in the relation between power consumption and resource utilization and can be classified as hardware- or software-centric (Dayarathna et al., 2016).

Models that follow complex approaches, i.e., by modeling the contribution of every hardware component and/or using machine learning methods for prediction are accurate, but they demand significant design and implementation effort. On the other hand, simple models, i.e., just based on overall server consumption and CPU utilization, demand significantly lower design and implementation effort, and can produce fairly accurate results. In the review of related work, the use of simple models has been predominant, especially until 2013 (see, for example Dayarathna et al. (2016) and Kurowski et al. (2013)).

Simulations are widely utilized for evaluating performance and energy models, schedule techniques, and others features of the execution of scientific workloads on real infrastructures (Hernández et al., 2014). Simulators allow assessing the evaluation of different alternative models and algorithms over realistic scenarios that capture the main features of nowadays computing infrastructures and applications. Using simulations, identical scenarios can be executed several times in order to perform statistical analysis. Moreover, simulators avoid the direct utilization of expensive hardware, allow a significant reduction in the execution time of the experiments (e.g., it is possible to simulate years of infrastructure utilization in just a few hours), and also permit considering a large number of scenarios, among other advantages (Bak et al., 2011). Due to the aforementioned reasons, cloud simulation is a key component of the research in the area. Many cloud simulators with different characteristics and specializations have been proposed in the literature (Malhotra and Jain, 2013). CloudSim is a simulation toolkit widely used in the literature. This tool allows modeling and simulating cloud computing infrastructures, provisioning environments, and applications (Calheiros et al., 2011). In this thesis, a custom version of CloudSim (Armenta-Cano et al., 2017) is extended by including new energy models, built considering the empirical power consumption evaluation, and different allocation policies.

In this line of work, this thesis focuses on the characterization of power consumption for applications over nowadays multicore hardware used in scientific computing platforms. Such characterization is useful for studying and understanding energy efficiency in data centers and for designing energy efficient scheduling strategies. Three synthetic benchmarks are studied over two physical servers from a real High Performance Computing (HPC) platform, registering their power consumption with a power meter device. Furthermore, the experimental analysis studies the power consumption of different applications sharing a computing resource via simultaneous execution. The proposed study is very relevant for nowadays data centers and HPC infrastructures that execute many applications, with an associated impact on both the energy efficiency and the quality of service (QoS) offered to the users of the platform. Furthermore, new energy models are built by applying polynomial regression on the empirical data. The new models are applied in simulation of data centers operation for predicting the power consumption and several scheduling strategies are evaluated in simulations performed considering real servers and scientific computing workloads.

The initial questions that have guided the research include the following: What is the relationship between the server load and its power consumption? Is power consumption different if the task is intensive in different computing resources? In that case, is it possible to save energy by executing tasks in the same server? What type of task is it convenient to execute together? To what extent is it possible to combine tasks without losing energy efficiency due to performance degradation caused by resource usage conflicts? What is the behaviour of the power consumption at the critical utilization level (about 100 % of server capacity)?

This thesis focuses on answering the aforementioned questions using an empirical approach. The experiments are designed to cover the domain of the problem and gather relevant information to process it in later stages using appropriate computational techniques.

The main contributions of this thesis are:

1. A cutting-edge review of related works regarding power characterization, modeling, and energy-aware scheduling in cloud computing and supercomputing systems. Also, a thorough review of available cloud simulation tools considering advantages and disadvantages of each one, is presented.
2. An empirical study of power consumption using benchmarks of the three main computing resources that most contribute to power consumption utilization (CPU, memory, and disk). The study considers computing resource isolated and combined, and different levels of server load. Two high-end multi-core servers (AMD and Intel architectures) are analyzed.
3. An empirical study of performance degradation in multi-core servers regarding the server load and the computing resource type. This informa-

tion is key in order to obtain scheduling strategies that provide accurate trade-off between power consumption and quality of service.

4. Several energy models built from experimental power consumption data using supervised computational intelligence techniques. Each model considers AMD and Intel architectures. Also, relevant metrics to asses the quality of each model are presented.
5. Energy evaluation through simulations for six scheduling strategies using realistic workloads.
6. The research follows the reproducible/replicable paradigm by using a Jupyter Notebook that shows in a clear and understandable manner the data processing from raw data. Notebooks are widely-used for sharing well-documented source code that can be executed easily. The reproducible/replicable paradigm allows to reduce errors and add new experiments data quickly. Also, the results and claims can be verified or extended by other researchers.

All research included in this thesis were published in two articles: a conference article and a special issue article in a journal. Next, each article is presented with a description of its content:

- Muraña et al. (2018) presented at *Latin American High Performance Computing Conference*, included the empirical analysis of power consumption in multicores. The study considered two high-end servers (AMD and Intel architectures) and benchmarks intensive in different computing resources. A performance measuring complemented the energy experiments.
- Muraña et al. (2019) published in *Cluster Computing* journal. The power and performance characterization study of the previous article were complemented with analytical models of power consumption and simulation results of scheduling algorithms. All results in this publication are publicly available to be verified from raw data through an ancillary Jupyter Notebook document available online.

Part of the research was performed in collaboration with Centro de Investigacion Cientifica y de Educacion Superior de Ensenada in Mexico (CICESE). The main concepts of this research were discussed in collaboration. Also, methodologies, artifacts, and state-of-the-art reports were exchanged mainly in early phases of the research. Some articles published by CICESE in relation to this collaboration were Armenta-Cano et al. (2017) and Galaviz-Alejos et al. (2018).

Two additional articles were published by the author of this manuscript in related areas (scheduling and QoS in HPC systems) in the beginning of the research.

- Muraña et al. (2014) presented at *2014 XL Latin American Computing Conference (CLEI)* included the application of a parallel evolutionary algorithm for solving a scheduling problem in cluster infrastructures. Two objectives were considered for optimization: makespan and tardiness. In addition, scheduling problems in heterogeneous computing and parallel evolutionary techniques were reviewed.
- Nesmachnow and Muraña (2014) presented at *III ALIO/EURO Workshop on Applied Combinatorial Optimization* included a comparison of an evolutionary algorithm based in linear aggregation and a multi-objective evolutionary algorithm to solve a scheduling problem in heterogeneous computing systems considering QoS objectives.

The thesis is organized as follows. Chapter 2 presents the motivation of the research and the main concepts of energy consumption/utilization in data centers. Chapter 3 reviews related works on energy characterization in multicores and simulators for energy-aware data centers. Chapter 4 presents the route map of the research. The different stages, their inputs and outcomes are introduced. Also, the proposed methodology for energy characterization, the benchmarks, and the physical setup for experiments, are described. The experimental evaluation of power consumption and performance of different benchmarks, is reported and discussed in Chapter 5. Chapter 6 describes the details of the power consumption models and performance models, built using polynomial regression. These models quality is assessed and compared by utilization of relevant metrics. In adition, several scheduling strategies for data centers, based on well-known heuristic are presented. The strategies were compared according to its energy efficiency through a simulation tool, using realistic workloads. Finally, the conclusions and main lines for future work are formulated in Chapter 7.

Chapter 2

Energy efficiency in data centers

This chapter describes the problem addressed, and the context and motivation of this research. Section 2.1 introduces energy efficiency as one of the key issues in data centers. The main characteristics of power consumption models and performance models in high-end server are presented in Section 2.2. Finally, Section 2.3 describes the scheduling problem in computing infrastructures, in particular, scheduling oriented to minimize power consumption.

2.1 Data centers and energy efficiency

This section presents an insight of energy efficiency in data centers. Subsection 2.1.1 describes the role of data centers today and why energy is an important concern. Also, details about the electricity market and the role of data centers are explained. Later, the section presents a general breakdown of data center power consumption. Well-known techniques oriented to improve data centers energy efficiency are described in Subsection 2.1.2.

2.1.1 Data centers and energy context

Nowadays, data centers are key components of IT services. Most of entertainment, scientific, and enterprise applications use data centers as background layer (Buyya et al., 2013). The current configuration of the IT industry, with a large-scale computational such as a backbone, has been driven by the increase of mobile users, software as a service as a business model, big data and IoT expansion, among others (Shi et al., 2016).

Cloud computing is a well established IT paradigm, where computing resources (storage, computing power, etc.) are created on-demand, in ubiquitous manner. The development of high-capacity networks and virtualization technologies has been key for the success of cloud computing (Buyya et al., 2013). Cloud data centers are large scale computational infrastructures that provides the hardware and software support for the cloud.

Cloud computing is a main paradigm in industry right now, and by 2021 94 % of the workload processed in data centers will be processed in cloud data center (Cisco Systems, 2016). The migration from the traditional data centers to cloud data centers indicates that the coupling between the logical and physical servers decreases. Energy efficiency techniques based on scheduling and reallocation acquire relevance in an scenario of complete virtualization, since they can be applied with less effort and cost than in a traditional scenario.

Data centers are big consumers of electricity. An average-sized data center consumes the same power of 25000 households (Beloglazov et al., 2012). Furthermore, in 2016, data centers worldwide consumed around of 3% of the whole electricity and this consumption will double every four years (Danilak, 2017). Due to this high consumption and its projection, the energy concerns of data centers and its environmental implications (Bawden, 2016) make data center energy management a current issue addressed by scientific research, government, and industry (Rong et al., 2016; Shehabi et al., 2016; Gao, 2014; Google LLC, 2019a).

The complexity of the electric power industry is increasingly driven by smart grids and emerging transportation technologies such as electric cars. In addition, the growing supply of renewable energy and the variation of its availability contribute to this complexity. Although the complexity makes more difficult power system planning, it also presents new business opportunities for large electricity consumers or groups of small electricity consumers to play different roles in the electricity market (Gungor et al., 2011; Moreno et al., 2012). One of these roles is to maintain the balance of supply and demand, increasing the power consumption when the supply of electricity is greater than the demand or reducing the power consumption when the demand is greater than the supply. The demand response agent (for example, the data center) must be able to reduce the power consumption of its operation (for example, by deferring jobs) or increase its power consumption for a given period of time. This role is important in modern power grids, since it avoids discarding electricity and also avoids the monetary cost of replacing damaged electrical equipment due to overload. The power management of data centers is a fundamental step for strategies to play these roles.

Two main operational components account for most power consumption of data centers: i) operation of the technological infrastructure (servers, network, storage, etc.) and ii) operation of the cooling system and other physical resources (Nesmachnow et al., 2014; Rong et al., 2016). Both sources of power consumption are related because more power is required for the cooling system when servers operate at full capacity. Servers represent a significant percentage of data center power consumption and the variability of their power consumption in different load levels allows implementing specific techniques for energy saving. This research focuses on reducing the power consumption of data centers due to operation of the technological infrastructure, by applying intelligent management of computing resources according to the main features of incoming workloads and applications.

In summary, the data centers are important users of the electricity network and for this reason their energy policies can have a high impact on the environment and a high monetary cost. Efficient energy management is key to minimize the environmental impact and to reduce monetary costs, and it is the first step to insert the data center into the electricity market, playing a more important role than that of a simple consumer.

2.1.2 Toward energy efficiency in data centers

Four main issues can be identified in order to increase energy efficiency of data centers (Rong et al., 2016): i) high-performance computing, which includes a set of software-centered techniques such as server/processor resources scheduling, network optimization, and compiler optimization ii) low-power servers, which includes energy optimization of hardware components, iii) energy conservation of computer rooms, and iv) utilization of renewable energy. This work mainly addresses issue i), and more specifically the server resources scheduling. However, the gathered characteristics of servers power consumption can be used to tackle the other issues too.

One of the predominant techniques in the literature to reduce servers power consumption is to scale the voltage and frequency of the CPU, i.e., dynamic voltage and frequency scaling (DVFS) (Le Sueur and Heiser, 2010). DVFS consists of decreasing or increasing the voltage and frequency of the CPU circuits. This way, the processing speed of the CPU decreases, consuming less energy than the normal state. The disadvantage of this decrease is the performance degradation. However, considering the tasks requirements, this degradation could be negligible. DVFS can be applied controlled by hardware or software (Mishra and Tripathi, 2014; Gade et al., 2015; Zhang and Hoffmann, 2016).

Another predominant technique in the literature to reduce the power consumption of servers is the energy-aware scheduling or relocation of tasks or virtual machines (Kaur and Chana, 2016; Fernandes et al., 2016; Chen et al., 2015). These techniques aim at consolidating tasks or virtual machines in the same server, in order to reduce the number of servers on idle state (i.e. turned-on server without load).

The downside of energy-aware policies is the degradation of QoS (Beloglazov et al., 2012). As mentioned above, the application of DFVS techniques or server overload due to the task reallocation can slow down tasks and provoke SLA violations. Considering both energy and performance objectives is mandatory in order to find appropriate trade-offs in the design of management policies. The first step to understand the power consumption of a server is to build a power model that synthesize the information of the power consumption phenomenon. Next section presents an insight into power consumption models of multi-core servers.

2.2 Modeling high-end multicore servers

This section introduces the concepts of power consumption modeling and performance modeling in high-end multicore servers. Subsection 2.2.1 presents a breakdown of the overall power consumption of a server. In particular, the model version implemented in this thesis is discussed. Performance models are presented in Subsection 2.2.2. Finally, Subsection 2.2.3 describes relevant metrics to assess the proposed models.

2.2.1 Power consumption modeling

Power consumption models allow predicting the power consumption of a given host when executing tasks. Such prediction is useful for several proposals, for example, for analyzing the energy efficiency of computing facilities via simulations. Power consumption of high-end servers that usually operate on data centers is broadly described by Equation 2.1 (Chen et al., 2013). This is the simplest approach to be considering regarding to the modeling of power consumption of a server. In the model, two components are identified: one component is static (P_{fix}) and the other is variable (P_{var}).

$$P_{server} = P_{fix} + P_{var} \tag{2.1}$$

The static component P_{fix} is the minimum power necessary to keep the the host operative. The static power is necessary because servers are composed by CMOS circuits, which need a minimum amount of energy to keep their state. The variable power P_{var} corresponds to the power consumption of the execution of the tasks.

The model presented in Equation 2.1 is extended in Equation 2.2. In this case, P_{fix} is renamed as P_{idle} (i.e., the server power consumption without load), and P_{var} is represented by the difference between the maximum power consumption of the system (P_{peak}) and the minimum (P_{idle}), multiplied by a function of the utilization ($f(u)$). The function $f(u)$ describes the relationship between utilization and power consumption. The variable u is the current utilization of the server (Beloglazov et al., 2010; Chen et al., 2013).

$$P_{server} = P_{idle} + (P_{peak} - P_{idle})f(u) \tag{2.2}$$

Several variables, such as server design and architecture, server utilization, and application type are related with the terms on Equation 2.2, and one of the main objectives of this thesis is to study this relationship.

Most of power consumption of servers corresponds to the CPU. However, power consumption of other computing resources (memory, disk, network) are not negligible. Feng et al. (2005) presented the following results in this regard, for server executing a memory-bound benchmark: 35% of the total server consumption corresponds to CPU, 16% to physical memory, and 7% to disk. The rest is consumed by power supply, network interface and other components.

It is possible to build highly accurate power consumption models, taking into account the type of computing resources in which the tasks are intensive. Equation 2.3 shows a server power model where u_{CPU} is the percentage of server capacity executing workload categorized as CPU-intensive, u_{mem} is the percentage of server capacity executing workload categorized as memory-intensive, and so on for each resource in the model.

$$P_{server} = P_{idle} + (P_{peak} - P_{idle}) f(u_{CPU}, u_{mem}, u_{disk}, u_{net}, \dots) \tag{2.3}$$

According to the literature reviewed, power consumption models can be classified on three categories: *static*, where the consumption of the host is a constant value; *dynamic*, where the value of the power consumption depends of the utilization level of the host; and *application specific*, where the consumption is associated to the characteristics of each application (Kurowski et al., 2013). Models with higher complexity can be built considering the competition for computing resources between those applications running at the same time in a host (Gao et al., 2014).

The power consumption models presented in this work are within the *application specific* type. Also, the competition for computing resources is considered. Models are built for estimating the overall power consumption as a function of the utilization of the host and type of workload executed (CPU, memory, or disk intensive). Models are built from experimental data of power consumption measurements in real hardware. This approach allows to consider the holistic behavior of the power consumption, which depends on multiple factors (chip design, voltage, cooling system, etc.). The experimental data is processed with data science tools and used as training information for machine learning modeling. Machine learning is a state-of-the-art technique driven by the exponential increase of data in the last decade and it is widely used for construction of statistical models (Bishop, 2006; Marsland, 2011; Bottou et al., 2018).

2.2.2 Performance modeling

Energy aware policies are in conflict with QoS (Beloglazov et al., 2012). Due to this conflict, it is necessary to complement the power consumption study with a performance analysis oriented to understand the QoS degradation caused by energy saving decisions. The complete information about power consumption and performance allows establishing service levels in the data centers, with different profiles such as energy efficiency levels, optimum levels, and levels with high benefits of performance. Performance models allows predicting how host load and job resource requirements affect performance metrics. Relevant performance metrics are the response time, the throughput, and the number of jobs in the system (Harchol-Balter, 2013). In addition, makespan (total completion time of the tasks of the scheduling) and tardiness (sum of the time when the tasks of the scheduling are executed after their deadline) are relevant metrics regarding to performance and QoS analysis (Muraña et al., 2014). The performance study in this thesis corresponds to the analysis of the degradation of multicores features when increasing the use of the cores, considering workloads intensive in different computing resources. Equation 2.4 shows the relationship between the completion time of batch tasks (i.e., tasks where user interaction is not required once processing begins) and the utilization percentage of the multicore system.

$$CT(b_1, ..., b_n) = f(n) \times CT(b_1) \tag{2.4}$$

In Equation 2.4, $CT(b_1, ..., b_n)$ is the completion time of n batch tasks (i.e. the makespan of the tasks set) and $CT(b_1)$ is the completion time in the minimum utilization percentage, i.e, one batch task executing alone in the server, using one core. The function $f(n)$ is the coefficient of the relation (it is greater than one). The empirical study reported in Section 5.2 corresponds to the model presented in Equation 2.4. Since the batch tasks studied in this research executes in one core with exclusivity, it is expected low degradation of the system when increasing the load. However, the access to shared resources as memory and disk causes degradation due to the competition (Dick and Mao, 2010; Zhuravlev et al., 2010). For this reason, it is important to consider different type of task when building the model.

2.2.3 Evaluating the quality of power consumption and performance models

Two relevant metrics to assess the quality of statistical models were applied to analyze the results: *coefficient of determination* (R-squared or R^2) and *adjusted R-squared* ($\bar{R}^2$). Both metrics evaluate the forecasting capabilities of the studied model (i.e., prediction of future values in a given temporal series). $\bar{R}^2$ is an extension of R^2 proposed to avoid spurious increasing when using a larger number of independent variables (Theil, 1961).

Figure 2.1 shows the intuitive idea of R^2, identifying two components: first SS_{tot} (Subfigure 2.1a), defined as the total sum of squared distance between each point of the data set and the average model; Second SS_{res} (Subfigure 2.1b), defined as the sum of squared distance between each point of the data set and the evaluated model. Equation 2.5 and Equation 2.6 present SS_{tot} and SS_{tot}, where n is the sample size, y_i are the measured values and $\bar{y}$ is the average model.

$$SS_{tot} = \sum_{i=1}^{i=n} (y_i - \bar{y}) \tag{2.5}$$

$$SS_{res} = \sum_{i=1}^{i=n} (y_i - f_i) \tag{2.6}$$

R^2, defined by Equation 2.7, express the improvement of the evaluated model regarding to the average model. In order to penalize the improvement of R^2 when the number variables increase, metric $\bar{R}^2$, is defined by Equation2.8, where p is the number of independent variables.

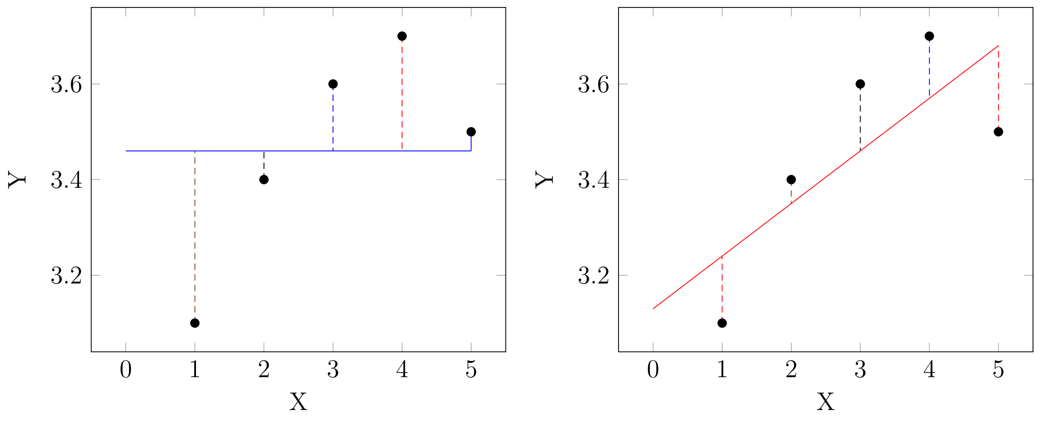
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2.7}$$

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{2.8}$$

2.3 Energy-aware scheduling

Scheduling in computing systems refers to assigning tasks to computing units following some criteria (minimized execution time, minimized energy consumption, etc.)(El-Rewini et al., 1994; Pinedo, 2016).

As in any other discipline, inadequate scheduling in the computer system can cause delays, waste of resources and a high monetary cost. In addition, in realistic scenarios, well-designed scheduling requires significant effort, since these are, in general, computationally complex problems (non-



(a) SStot **(b)** SSres

Figure 2.1: Graphical representation of the R-squared components

polynomial). Due to this complexity, scheduling problems are usually addressed using heuristic, meta-heuristic and ad-hoc solutions.

The aforementioned concepts make computing system scheduling a widely addressed problem by scientific community (Jiang et al., 2007; Zhan et al., 2015).

Energy-aware scheduling in computing system is the application of scheduling techniques to reduce energy consumption. The formulation of the energy optimization problem, without considering QoS, is to find a schedule that minimizes the value E_{tot} of the expression presented in Equation 2.9. E_{tot} is the total energy required to complete the tasks execution. There are N computing units (servers) and M tasks to be assigned. The function e corresponds to the energy consumed by task j executing in server i.

$$E_{tot} = \sum_{i=1}^{N} \sum_{j=1}^{M} e(i, j) \tag{2.9}$$

In non-preemptive scheduling problems (Sousa and Wolsey, 1992), such as the one addressed in this thesis, function e of a task has a non-zero value only in one server within a schedule (this is, a task is assigned only to one server). The function e depends on several factors, such as the server, the task, and moreover, it depends on the characteristics of the current load of the server.

The same energy-aware scheduling problem is illustrated by Equation 2.10, this time, from the point of view of the servers power consumption. This server approach simplifies the coding of the simulation and is the one used in the case study presented in Section 6.3. The function u_i is the utilization percentage of the server i at time t and function P_i is the instant power consumption of server i. T is the duration of the scheduling period.

$$\sum_{i=1}^{N} \int_{t=0}^{t=T} P_i(u_i(t)) \tag{2.10}$$

Equation 2.11. corresponds to the discretization of the time T in K intervals of time, in which the utilization servers is constant. The duration of the time k is d_k. This way, to know the total energy consumption of a schedule, it is necessary to know P_i and the utilization percentage of each server in each time k.

$$\sum_{k=0}^{K} \sum_{i=1}^{N} P_i(u_{ki}) \times d_k \tag{2.11}$$

As mentioned in previous sections, power consumption function depends on the type of task. For this reason, the value u_{ki} corresponds to a vector of the form $[u_{CPU}, u_{mem}, u_{disk}, ...]$.

Chapter 3

Related Work

This chapter presents the review the related literature. Section 3.1 presents the revised works related to characterization and modeling of power consumption of servers and energy-aware scheduling. Then, Section 3.2 presents a comprehensive review of the state-of-art in relation to cloud simulation tools. Finally, Section 3.3 summarizes the reviewed works and presents the conclusions of the analysis.

3.1 Power characterization and energy-aware scheduling review

Iturriaga et al. (2014) studied the multiobjective optimization problem to optimize power consumption and execution time in heterogeneous computers systems, considering uncertainty. Specific versions of well-known heuristic were proposed for scheduling on realistic scenarios, applying the power consumption model defined in Nesmachnow et al. (2013) and considering only CPU-bound workloads. A model for uncertainty on power consumption was determined through empirical evaluations using three CPU-bound benchmarks. Regarding scheduling results, online heuristics computed better schedules than offline approaches. Results also confirmed that uncertainty has a significant impact on the accuracy of the scheduling algorithms. The power consumption behavior of CPU-bound benchmarks shown by Iturriaga et al. (2014) is consistent with the one reported in our research. Moreover, we propose a fully empirical power consumption characterization, also considering two additional types of benchmarks: memory bound and disk bound.

Srikantaiah et al. (2008) studied workload consolidation strategies for energy optimization in cloud computing systems. An empirical study of the relationship between power consumption, performance, and resource utilization was presented. The experiments were executed on four physical servers connected to a power meter to track the power consumption, and resource utilization was monitored using the Xperf toolkit. Only two resources were considered in the study: processor and disk. The performance degraded for high levels of disk utilization, and variations in CPU usage did not result in significant performance variations. Energy results were presented in terms of power consumption per transaction, resources utilization, and performance degradation. Results also showed that power consumption per transaction is more sensitive to CPU utilization than disk utilization. The authors proposed a heuristic method to solve a modified bin packing problem where the servers are bins and the computing resources are bin dimensions. Results reported for small scenarios showed that power consumption of the solutions computed by the heuristic is about 5% from the optimal solution. The tolerance for performance degradation was 20%.

Du Bois et al. (2011) presented a framework for creating workloads with specific features, applied to compare energy efficiency in commercial systems. CPU-bound, memory-bound, and disk-bound benchmarks were executed on a power monitoring setup composed of an oscilloscope connected to the host and a logging machine to persist the data. Two commercial systems were studied: a high-end with AMD processors and a low-end with Intel processors. Benchmarks were executed independently, isolating the power consumption of each resource. Results confirmed that energy efficiency depends on the workload type. Comparatively, the high-end system had better results for the CPU-bound workload, the low-end system was better for disk-bound, and both had similar efficiency for the memory-bound workload. Our work complements this approach by including a study of the power consumption behavior when executing different types of tasks simultaneously on specific architectures for high performance computing.

Feng et al. (2005) evaluated the energy efficiency of a high-end distributed system, with focus on scientific workloads. The authors proposed a power monitoring setup that allows isolating the power consumption of CPU, memory, and disk. The experimental analysis studied single node executions and distributed executions. In the single node experiments, results of executing a memory-bound benchmark showed that the total power consumption is distributed as follow: 35% corresponds to CPU, 16% to physical memory, and 7% to disk. The rest is consumed by the power supply, fans, network, and other components. Idle state represented 66% of the total power consumption. In distributed experiments, benchmarks that are intensive in more than one computing resource were studied. Results showed that energy efficiency increased with the number of nodes used for execution.

Kurowski et al. (2013) presented a data center simulator that allows specifying various energy models and management policies. Three types of theoretical energy models are proposed: i) *static approach*, which considers a unique power value by processing unit; ii) *dynamic approach*, which considers power levels, representing the usage of the processing unit; and iii) *application specific approach*, which considers usage of application resources to determine the power consumption. Simulation results were compared with empirical measurements over real hardware to validate the theoretical energy models in arbitrary scenarios. All models obtained accurate results (error was less than 10% with respect to empirical measurements), and the dynamic approach was the most precise.

Langer et al. (2015) studied energy efficiency of low voltage operations in manycore chips. Two scientific applications were considered for benchmarking over a multicore simulator. The performance model considered for a chip was $S = a_k(\sum f_i) + b_k$, where S are the instructions per cycle, f_i is the i-th core frequency and a_k,b_k are constants that depend on k, the number of cores in the chip. A similar model is used for power consumption. Across 25 different chips, an optimization method based on integer linear programming achieved 26% in energy savings regarding to the power consumption of the faster configuration.

There are different opinions in the related literature about the importance of network power consumption. On the one hand, Feng et al. (2005) measured the power consumption of each computing resource in isolated manner using different scientific computing benchmarks, and network turned to be in the fourth place in terms of power consumption, behind CPU, memory, and disk. Moreover, a recent survey by Dayarathna et al. (2016) presented a graphic distribution of power usage by components, based on results from the analysis of a Google data center by Barroso et al. (2013), where network power consumption is behind the other computing resources. A comparison between two Intel hosts (with Xeon and Atom processors), based on results from Malladi et al. (2012), showed that the network power consumption is less than CPU, memory, and disk consumption. On the other hand, Totoni et al. (2013) presented a runtime power management for saving energy by turning on/off not used or underutilized links during the execution of applications. The authors justified the relevance of the study, regarding the power consumption of the system, based on the use of low frequency many cores in computing systems and the complexity of network design nowadays. However, the study was not specifically aimed at analyzing scientific workloads.

3.2 Cloud simulation tools

Kurowski et al. (2013) proposed DCworms, an event-driven data center simulator written in Java, which allows defining performance models, energy models, and scheduling polices. DCworms is built using GSSIM, a Grid simulator by Bak et al. (2011). Among other features, GSSIM allows simulating a variety of entities and distributed architectures. Regarding to power consumption, GSSIM provides several energy models classified in *static*, *resource load* and *application specific*. In the third type of model, the user can specify energy profiles that allow modeling the type of application and the type of resource. Energy information is logged for analyzing the simulation.

Calheiros et al. (2011) introduced CloudSim, a cloud simulator developed in Java. Among the main advantages of CloudSim are the virtualization layer, which allows defining an abstraction of different execution environments of the applications in virtual machines, and the support provided for implementing federated clouds, which allows modeling large distributed systems. The simulator provides functions for implementing custom energy models. In addition, new scheduling polices can be included, for instance the strategies oriented to optimize the power consumption developed in our work. The total power consumption of a simulation can be used to compare the efficiency of scheduling policies.

Kliazovich et al. (2010) presented GreenCloud, a simulator oriented to energy-aware data centers. The GreenCloud design allows measuring the power consumption of each component of the infrastructure (host, switch, etc) in a detailed manner. A set of energy efficiency strategies are provided, including DVSF and dynamic shutdown of the computing components. These GreenCloud characteristics allow implementing fine-grained energy-aware scheduling strategies.

Núñez et al. (2012) introduced Icancloud, a Cloud Simulator which allows reproducing Amazon EC2 instances types. Also, it is able to customize the VM brokering due to the flexible design of the hypervisor model. The hosts can be single core or multicore, and the software is able to simulate long time periods (years). Regarding storage, local devices, NFS, and parallel storage can be simulated.

3.3 Summary

Table 3.1 summarizes the related work reviewed, including a description of the main characteristics of the work.

Power characterization and energy-aware scheduling	
reference	*description*
Iturriaga et al. (2014)	Studied the power consumption of several CPU-bound benchmarks.
Nesmachnow et al. (2013)	Definition of a power consumption model only CPU-bound workloads..
Srikantaiah et al. (2008)	Studied of the relationship between power consumption, performance, and resource utilization. Proposed an heuristic to solve energy-aware scheduling problem as a modified bin packing problem.
Du Bois et al. (2011)	Studied the power consumption of CPU-, memory-, and disk-bound benchmarks using a hardware-centered power monitoring setup. Included a comparison between high- and low-end servers.
Feng et al. (2005)	Proposed a power monitoring setup to evaluate the energy efficiency of a high-end distributed system. The setup allowed isolating the power consumption of CPU, memory, and disk.
Kurowski et al. (2013)	Evaluated an energy-aware scheduler by comparing with empirical results over real hardware. Results showed an error lower than 10 % with respect to the empirical measurements.
Langer et al. (2015)	Studied energy efficiency of low voltage operations in manycore chips, considering scientific applications.
Cloud simulation tools	
reference	*description*
Kurowski et al. (2013)	Proposed DCworms, including as main features the ability of defining performance models, energy models, and scheduling polices. Also, the energy information is logged.
Calheiros et al. (2011)	Introduced CloudSim, including as main features a virtualization layer, functions for implementing custom energy models, and the ability of including fine-grained energy-aware scheduling strategies.
Kliazovich et al. (2010)	Presented GreenCloud, including as main feature a detailed power measurement of each component (hosts, switches, etc)
Núñez et al. (2012)	Introduced Icancloud, including as main features a customizable VM brokering, simulations of long periods, and the ability to define single or multi-core host.

Table 3.1: Summary of the related work

Several works in literature have focused on modeling and characterizing power consumption of scientific applications. However, to the best of our knowledge, there is no empirical research focused on the inter-relationship between power consumption and CPU, memory, and disk utilization. Also, there is no experimental analysis of critical levels of resource utilization (close to 100%) and its impact on power consumption and performance. This thesis contributes to this line of research, proposing an empirical analysis for both issues mentioned above.

The review allows identifying several desirable features of the available cloud simulators that are useful for the analysis reported in this thesis, including support for virtualization, single and multicore hosts, distributed systems, energy and performance models available, scheduling policies available, ease of use, and flexible customization. CloudSim includes many of the aforementioned useful characteristics and also has a detailed documentation. For these reasons, we selected CloudSim as the simulator to use in the power consumption evaluation reported in this thesis.

Chapter 4

Methodology for power consumption and performance evaluation

This chapter describes the proposed methodology for the study of power consumption. Section 4.1 presents the workflow that describes the energy optimization process of data centers operation, addressed in this thesis. The section also describes the goal of each step of the workflow. Section 4.2 introduces a general description of power characterization and the experiments details such as benchmarks, hosts and design decisions. Finally, Section 4.3 describes the reproducible/replicable research paradigm and the software tools used in this thesis to follow it. The section also presents an exhaustive description of the data extraction and modeling.

4.1 Energy optimization workflow

Several steps are identified in the process of obtain efficient energy-aware policies with QoS trade-off (Dayarathna et al., 2016). Figure 4.1 shows the workflow that includes these steps and its relations. Although the steps are related, their outcomes can be used independently, for instance, conclusions obtained in the power characterization step can be applied in future modeling works or directly as input of scheduling heuristics design. Power characterization step aims at analyzing the behaviour of power consumption in different load conditions and computing resource usage. Performance characterization step studies the system degradation under the same conditions described for the power consumption characterization. Both characterization steps are totally empirical and their outputs are: i) numeric measurements data, i.e., raw information that is the main input for applying machines learning techniques in modeling stages and ii) graphics and observations used for extracting patterns that can lead to the development of scheduling heuristics.

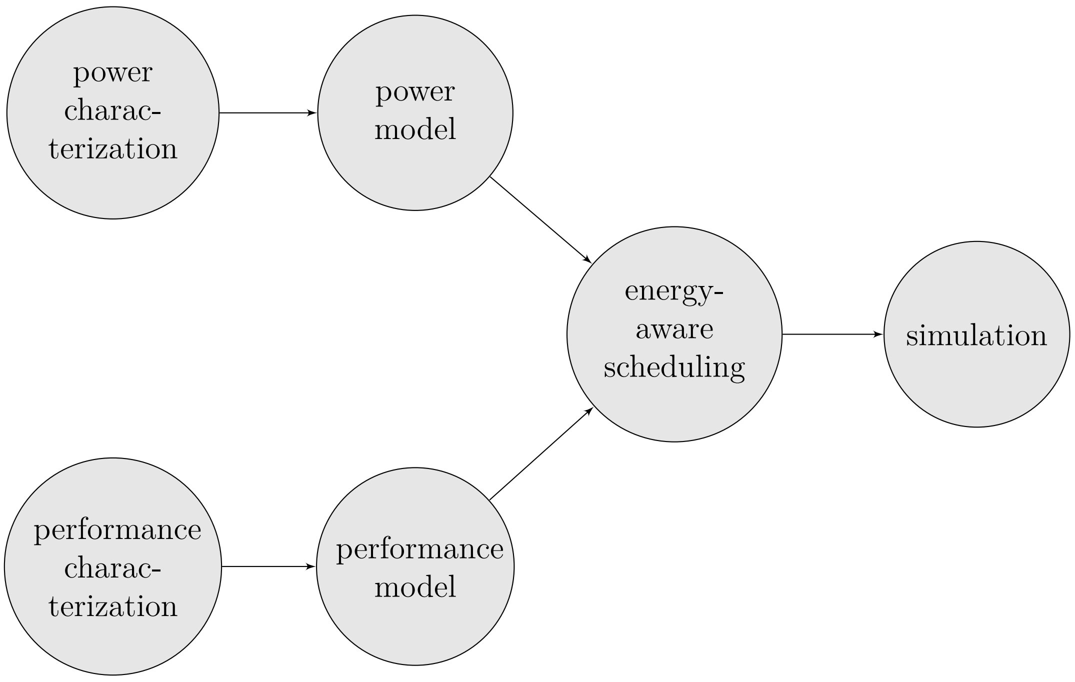


Figure 4.1: Workflow of proposed methodology

In modeling steps, the experimental data is adjusted to functions in order to predict future values. The obtained functions allow synthesizing the behaviour of the registered phenomenon (in this case, power consumption and performance of the benchmarks) in a simpler expression. Energy-aware scheduling step consists in design scheduling strategies aimed at reducing power consumption. For the design, well-known heuristic strategies applied in scheduling problems (Min-Min, Max-Min, etc.) are complemented with information extracted from characterization step, in order to build new heuristics. The last step of the optimization process corresponds to the simulation, and it allows to evaluate and compare the scheduling strategies. Simulation is an important tool in complex optimisation scenarios like data center scheduling, and it is widely-used in research area. The use of simulation tools instead of ad hoc software to evaluate scheduling strategies saves time because of the reuse of algorithms from the research community and avoids possible errors in the implementation.

4.2 Power characterization

This section describes the general design of the power characterization experiments. Subsection 4.2.1 presents an overview of power and performance characterization. Subsection 4.2.2 describes the benchmarks considered in the study and Subsection 4.2.3 introduces the details of the hardware used in the experiments. Then, Subsection 4.2.4 presents a thorough description of experiments design, and defines relevant nomenclature for the rest of the manuscript.

4.2.1 General overview

The experiments design was oriented to characterize the power consumption of the most important computing resources from the point of view of energy efficiency: CPU, memory and disk (Barroso et al., 2013; Du Bois et al., 2011; Feng et al., 2005; Iturriaga et al., 2014; Malladi et al., 2012). The importance of network power consumption has increased in the last years, mainly in modern data centers that offer distributed services, which heavily rely on network communications (Zhang and Ansari, 2015; Guo et al., 2016). However, network power consumption analysis was not included in the characterization, due to two main reasons: i) scientific computing applications in multicore architectures, on which the study of this thesis is centered, do not necessarily have a great use of the network. In these applications, the multithreaded paradigm allows solving complex scientific computing problems, without using the network extensively, since they mainly apply shared memory communication methods and ii) the complexity of the experiments, since to performing measures of network power consumption implies the consideration of several components such as switches, network interfaces, network topology, and cards, etc. Furthermore, the relevance of network power consumption is subject of debate, as we already acknowledged in the review of related literature in Section 3.1.

The main goal of the analysis was studying the holistic behavior of the power consumption of a host, considering the following elements, which are correlated: the utilization level of each resource; the total utilization level of the host; and the types of resources involved. In consequence, experiments were designed to execute synthetic benchmarks that make intensive utilization of different computing resources, which allow capturing the features of different scientific computing applications in multicore computers. Benchmarks were executed in both isolated and combined manner, at different levels of resource utilization. Analyzing the power consumption of hosts at levels close to maximum utilization (100%) has received low consideration in related bibliography, thus it is included as one of the main relevant issues to study. Thus, all type of experiments performed consider the critic level of utilization.

Since the downside of reducing energy consumption is the degradation of system performance, it is necessary to complement the characterization of energy consumption with performance experiments. In addition to the relationship between system load and power consumption, the relationship between system load and execution time is studied. This way, it is possible to make trade-off decisions according to the context demand, such as energy prices, service levels, etc. In performance experiments, the same benchmarks were executed with a fixed computational effort as stopping criterion, instead of the wall-time considered in power consumption experiments. For example, the criterion for stopping the CPU-intensive benchmark is when the loop counter to find prime numbers is greater than 20000.

4.2.2 Benchmarks for power consumption characterization

Simple benchmarks (i. e., benchmarks that are intensive in a single computer resource) were used for the analysis, in line with several articles reviewed in the literature related to the evaluation of power consumption. For example, Iturriaga et al. (2014) considered a power consumption evaluation of LINPACK (Dongarra, 1988), a well-known CPU-intensive benchmark used for ranking supercomputers of the TOP500 list. The same benchmark were used by Kurowski et al. (2013), together with Abinit (Gonze et al., 2009), a software to calculate properties of material within density functional theory (which can be CPU or memory intensive depending on the configuration of parameters), NAMD (Phillips et al., 2005), a CPU-intensive software for biomolecular system simulation, and CPUBURN (Nielsen, 2012), a software that allows stressing the CPU. The approach that uses simple benchmarks was followed since analysis is oriented to characterize power consumption regarding the utilization of each computing resource, both isolated and when combining the utilization of several resources. For this reason, specific programs with intensive utilization of each one of the three studied computing resources (CPU, memory, disk) were needed. The chosen synthetic benchmarks allow isolating the utilization of each resource, in order to perform the characterization.

A set of benchmarks included in the Sysbench toolkit (Kopytov, 2017) were used in the analysis and characterization of power consumption. Sysbench is a cross-platform software written in C that provides CPU, memory, and disk intensive benchmarks for performance evaluation. The components used in the experiments include a CPU-bound benchmark, a memory-bound benchmark, and a disk-bound benchmark. The main details of each benchmark are described next.

CPU-bound benchmark. The CPU-bound benchmark is an algorithm that calculates $\pi(n)$ (the prime counting function) using a backtracking method. The algorithm in the benchmark includes loops, square root, and module operations, as described in Algorithm 1.

Algorithm 1 CPU-bound benchmark program

```
1: $c \leftarrow 3$
2: while $c <$ MaxPrime do
3:     $t \leftarrow sqrt(c)$
4:     $l \leftarrow 2$
5:     while $l < t$ do
6:         if $c \pmod t = 0$ then
7:             break
8:         end if
9:         if $l > t = 0$ then
10:            $n \leftarrow n + 1$
11:        end if
12:        $l \leftarrow l + 1$
13:    end while
14:    $c \leftarrow c + 1$
15: end while
16: return $n$
```

Memory-bound benchmark. The memory-bound benchmark is a program that executes write operations in memory, as described in Algorithm 2, where the *buf* variable is an array of integers. The cells of the array are overwritten with the value of *tmp* until the last position of the array, i.e., the value of the *end* variable.

Algorithm 2 Memory-bound benchmark program

```
1: while $buf < end$ do
2:     $*buf \leftarrow tmp$
3:     $buf \leftarrow buf + 1$
4: end while
```

Disk-bound benchmark. The disk-bound benchmark is a program that reads/writes content in files. Read or write requests are generated randomly and executed until a given number of requests (*MaxReqs*) is reached, as described in Algorithm 3.

Algorithm 3 Disk-bound benchmark program

```
1: while $ReqCount < MaxReqs$ do
2:     $Req \leftarrow generate\_rnd\_request()$
3:     if $is\_read(Req)$ then
4:         $read(Req.file)$
5:     end if
6:     if $is\_write(Req)$ then
7:         $write(Req.file)$
8:     end if
9:     $ReqsCount++$
10: end while
```

4.2.3 Multicore hosts and power monitoring setup

Experiments were performed on high-end servers from Cluster FING, the HPC and scientific computing platform from Universidad de la República, Uruguay (Nesmachnow, 2010). Two hosts were chosen according to their features and availability: HP Proliant DL385 G7 server (2 AMD Opteron 6172 CPUs, 12 cores each, 72 GB RAM), and HP Proliant DL380 G9 server (2 Intel Xeon E5-2643v3 CPUs, 12 cores each, 128 GB RAM). The considered hardware testbed covers the two most important architectures for high-end computers and high performance computing data centers nowadays. Table 4.1 presents the specification of both hosts.

To measure the power consumption of a server, two approaches are found in literature: software-based metering (i.e., in-band) and hardware-based metering (i.e., out-of-band) (Dayarathna et al., 2016; Grant et al., 2017). Software-based meters estimate the power consumption of some server components by consulting internal counters (Isci and Martonosi, 2003). An example of a

feature	*host server*	
	AMD	Intel
model name	AMD Opteron Processor 6172	Intel Xeon CPU E5-2643 v3
architecture	x86_64	86_64
cores	24	24
CPU frequency	2.1 GHz	3.4 GHz
threads per core	1	1
cores per socket	12	12
sockets	2	2
NUMA nodes	4	2
total memory	70 GB	126 GB

Table 4.1: Specification of servers used in experiments

software-based meter is *likwid* (Treibig et al., 2010), which reads the counter Running Average Power Limit (RAPL) of Intel architectures. On the one hand, software-based approaches have low cost and high scalability and, on the other hand, the power measurement is an estimate and is partial, since the CPU counters do not consider the consumption of server components such as fans, motherboard and others. Besides, counters and sensors are not available in many high-end server models, so the software-based approaches are restricted to one group of servers. Hardware-based approaches (Rashti et al., 2015; Laros et al., 2013) (the server is connected to an external power meter, which is connected to the power outlet) have the advantage that the power measurement is independent of the server model. Also, since the total energy consumption is considered, the measurements are accurate. On the downside of hardware-based approaches, the economic cost is greater than the measurement of the software and can not be scaled, since including a new host for power monitoring involves buying and installing new power meters. In addition, power meters installation is not possible if physical access to the IT equipment is restricted.

Because this thesis focuses on the overall power consumption of the server and studies its holistic behavior, the use of external power meters is more appropriate than software-based meters. Figure 4.2 presents the power monitoring setup applied in the research developed in this thesis. The presented setup allows reducing the measurements noise, since the processing that is not related to the benchmark executions (polling, log writing, etc.), is executed in an external machine.

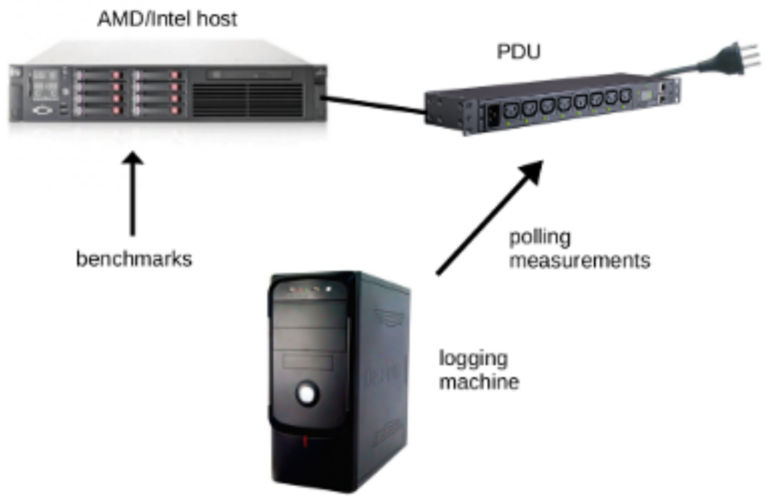


Figure 4.2: Power monitoring setup

The applied setup is similar to the one used in related works (Iturriaga et al., 2014; Du Bois et al., 2011; Srikantaiah et al., 2008). Benchmarks were executed in a host connected to the power source via a Power Distribution Unit (PDU) to register the instant power consumption. The PDU used is CyberPower PDU20SWHVIEC8FNET model. The PDU allows accessing the power consumption indicators through a web service. In an secondary machine, a polling demon logged data for post-processing. The complete description of the post-processing is presented in Section 4.3.2.

4.2.4 Design of experiments

In order to consider the general remarks of the key steps in the design of experiments suggested by Cox and Reid (2000), the design of the experiments in this thesis is a consequence of initial questions about the factors that affect the host PC of multi-cores (see Introduction 1). In the experiments, the average power consumption (PC) of each host was computed considering the measures obtained in 20 independent executions of each benchmark (in single benchmark experiments) or combination of benchmarks (in the combined benchmarks experiments), to obtain statistically significant values. Average and standard deviation of PC are reported for each benchmark in the experimental analysis. The average idle power consumption (IC), i.e., the average consumption of the host when it is not executing any workload, is computed considering the measures for 20 independent executions of a program with null operations. Both PC and IC are used to to compute the effective consumption (EC) for a given benchmark or workload as $EC = PC - IC$.

In order to consider an incremental approach to the experiment design that facilitates understanding, the experiments are described in three stages: single experiments, combined experiments and performance experiments. The stages are described in the next paragraphs.

First stage: single benchmarks. In a first set of experiments, benchmarks were evaluated isolated (i.e., independently from each other, analyzing only one resource). Utilization level (UL) is defined as the percentage of processors being used regarding the total number of processors in the host. Eight ULs were considered for single benchmark experiments: 12%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, and 100%. Figure 4.4 shows an example of 37.5% UL: the host has 24 processors of which 9 execute instances of the CPU-bound benchmark. The remaining processors are in *idle* state.

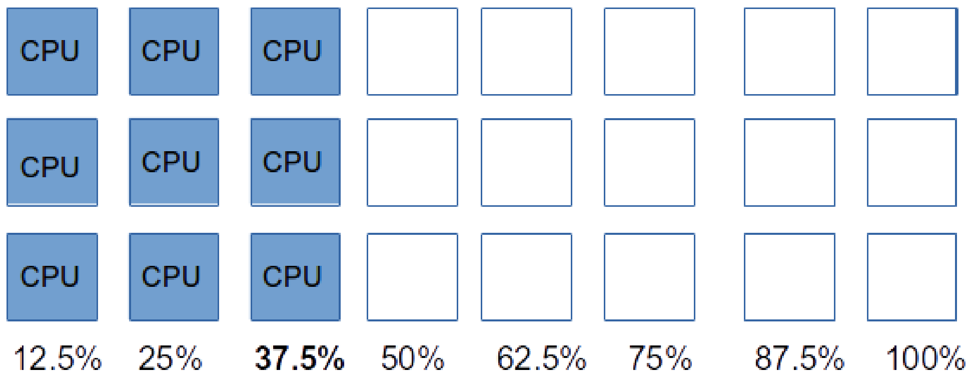


Figure 4.3: CPU-bound benchmark, UL of 37.5%

Second stage: combined benchmarks. In a second stage, the simultaneous execution of benchmarks was evaluated, analyzing several combinations of resource utilization at the same time. Two and three benchmarks were executed together in different ULs. In this case, UL is a vector where each entry represents the percentage of processors being used by each type of benchmark considered in the study (CPU-bound, memory-bound, and disk-bound, in that order). The chosen ULs were (25%,25%), (25%,50%), (25%,75%), (50%,25%), (50%,50%), (75%,25%) in pair executions, and (25%,25%,25%), (25%,25%,50%), (25%,50%,25%), (50%,25%,25%) in triple executions. The considered ULs cover the domain of possible combinations with a suitable granularity. Figure 4.4 shows CPU-bound and memory-bound executing together with UL (25%,50%) and Figure 4.5 shows the three benchmarks executing combined with UL (25%,25%,50%).

Each instance of the memory-bound benchmark was configured to use $(100/N)$ percent of the available memory, where N is the number of processors of the host. This configuration allows using 100% of the memory in full utilization mode. Each instance of the disk benchmark was configured to use 4 GB of disk size in AMD experiments and 2 GB of disk size in Intel experiments. These disk sizes were chosen taking into account the available disk capacity in each host. Instances were executed and monitored for 60 seconds. These duration allows ensuring the steady state of the benchmark execution, avoiding biasing the measurements to the initialization stages.

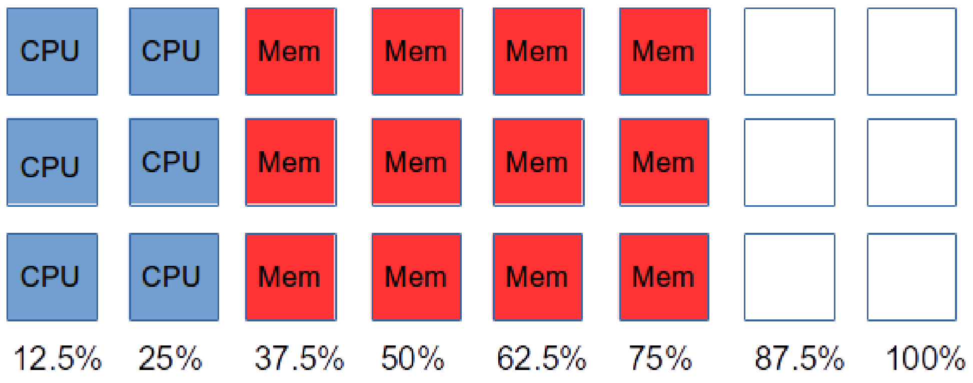


Figure 4.4: CPU- and memory-bound benchmarks combined, UL of (25%, 50%).

Third stage: performance evaluation. Finally, the impact on performance was analyzed. One of the most relevant metrics in performance analysis of applications is the makespan, defined as the time spent from the moment when the first task in a batch or bag of tasks begins execution to the moment when the last task is completed (Leung et al., 2004). In performance evaluation experiments, the makespan when executing multiple applications at the same time is compared with the makespan of single executions. In all cases, the average makespan values computed over 20 independent executions

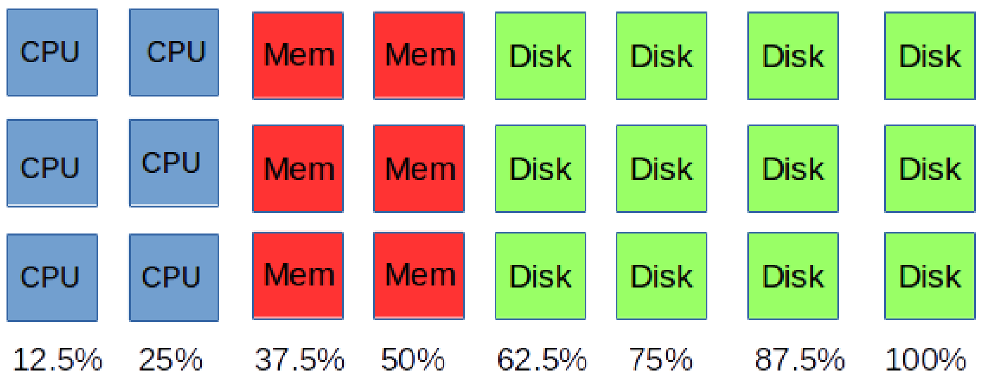


Figure 4.5: CPU-, memory-, and disk-bound benchmarks combined, UL of (25%,25%,50%).

of each benchmark or combination of benchmarks are reported, in order to provide statistical significance.

4.3 Processing data under reproducible/replicable research paradigm

This section describes the reproducible/replicable paradigm and its application to the research of this thesis. Also, the section presents an exhaustive description of the data processing and the models construction. Subsection 4.3.1 presents the definition of the main concepts of the paradigm and the motivation for its application. Subsection 4.3.2 presents a detailed description of the procedure of data processing, from the raw data to understandable tables and graphics. Finally, the details of models implementation using Python are presented in Section 4.3.3.

4.3.1 Reproducible and replicable research by using Jupyter Notebook and Pandas

Nowadays, reproducibility/replicability are fundamental characteristics in all scientific research areas (Patil et al., 2016; Repko, 2008). There are different definitions of these terms in literature (Plesser, 2018; Goodman et al., 2016). In this thesis, these concepts are defined as follows, based on guidelines of Association for Computing Machinery, Artifact Review and Badging (ACM, 2018).

Replicability: a study is replicable if the same measurements can be obtained with the same experimental setup by different researchers.

Reproducibility: a study is reproducible if the same measurements can be obtained with different experimental setup by different researchers.

The main ideas behind reproducible and replicable concepts are transparency and trust on research claims. Also, reproducible/replicable research allows bug detection and improvement by pairs. In addition, the clarity in the auxiliary artifacts of the research (raw data, scripts, etc.) allows a quick correction of errors and a collaborative work. In this thesis, in order to follow the paradigm of reproducible and replicable research, all raw data, processing tools, and processed results are provided for verification by researchers and practitioners. The design decisions are explained and the process of analysis is clearly documented. The published Jupyter Notebook allows reproducing the whole data processing and modeling, making easy correcting errors as well as extending the research by considering new experimental data and/or new models.

Jupyter Notebook (Jupyter Community; Kluyver et al., 2016) is a web application that allows integrating live programming code and documentation in markdown format. The combination of code and styled text is a powerful tool for presenting and sharing research results. Important industries such as Google and IBM have developed projects similar to Jupyter, especially oriented to data science applications. (Google LLC, 2019b; IBM, 2019).

The use of Jupyter Notebook and the transparency of the data processing are important contributions of the reported research, since the reproducible/replicable approach is not found often in related research areas (Begley, 2013). The complete data processing and modeling is publicly available in a Jupyter Notebook published in `https://www.fing.edu.uy/~jmurana/msc`.

The general description of the data processing tasks documented in the Jupyter Notebook are enumerated below.

1. Collect logs, clean data, and correct formats: log files, collected from log machine and hosts, are read and loaded on data structures. The data is cleaned by removing inconsistencies and parsed to appropriate data

types (e.g., a timestamp from string format to date format).

2. Combining information: the information of different sources is combined to obtain the required measurements.
3. Results analysis and plotting data: results are processed and analyzed to recognize patterns in order to characterize the power consumption.
4. Modeling and evaluation of the models: energy models are generated from data structures using linear and polynomial regression. The models are evaluated using statistical tests.

The data processing was preformed using Pandas (McKinney, 2011) and NumPy (Oliphant, 2006), among other Python libraries for the analysis of scientific data, which are oriented to high performance and ease to use.

Python is an open-source, interpreted, programming language oriented to readability and high performance (Oliphant, 2007). Python is a general-proposed language, however, it usage in data science applications has been notorious in the last years and it has becomes in de facto programming language in research areas related to artificial intelligence, big data, Internet of Things, among other edge technologies (Millman and Aivazis, 2011). Several programming paradigms, including object oriented, imperative, procedural, and functional are supported by Python. Also, the language provides powerful methods for list manipulation and an extensive standard library that provides potent modules to solve problems in a few lines of code. For the aforementioned characteristics, Python allows high productivity and maintenance.

Pandas is a Python data analysis library (McKinney, 2011). Pandas provides structures and tools for data analysis and modeling. The main structure of the library is the dataframe, which allows manipulating indexed data by rows and columns. Complex indexing, such as multi-columns, are handled too. The data can be loaded into a dataframe (or written to) directly from file in format CSV, XML, and others. The library also provides intelligent handling of missing data. Aggregation, merging, and joining are other operations supported by Pandas. In addition, it is possible to plot the content of dataframes content, which in combination with the integrated plotting of Jupyter Notebooks, allows a quick and intuitive visualization of the results.

4.3.2 Extraction of useful information from raw data

In order to accomplish with the aforementioned characteristics of a reproducible research, this subsection presents an exhaustive description of the energy and performance data processing.

Energy data extraction procedure The procedure applied to transform the raw data obtained from the experiments into useful information, described in Figure 4.6, it is implemented using Pandas library (McKinney, 2011) .

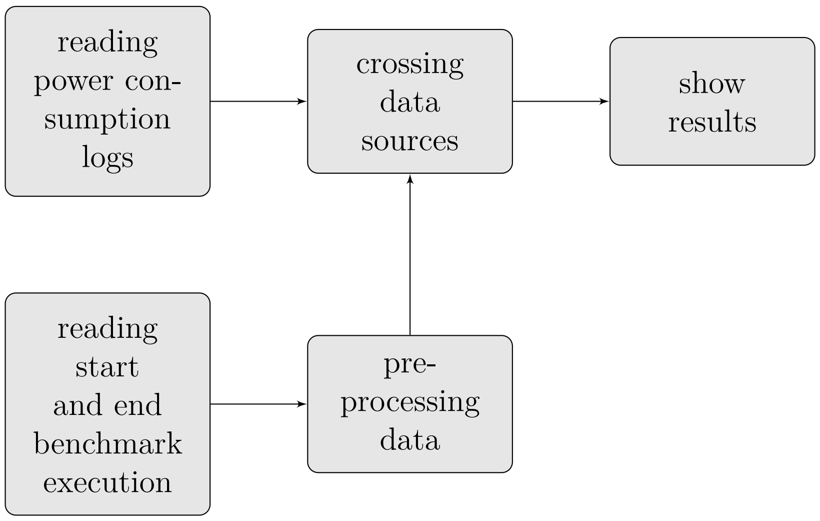

reading power consumption logs
crossing data sources
show results
reading start and end benchmark execution
pre-processing data

Figure 4.6: Procedure for energy data extraction

The procedure of data extraction involves the consolidation of two data sources, which correspond to the first two stages: reading of power consumption logs (written in the logging machine) and reading of start and end of benchmark executions (written in hosts). By comparing the datetime of records from both sources, the energy consumed by the execution of a benchmark is computed. The rest of the stages are: pre-processing data, crossing data sources, and show results.

Reading power consumption logs. The information is obtained by polling every second the power meter and logging the instant power measured in a text file. An example of the format of the power consumption logs is presented in Listing 4.1. The relevant fields for measuring power consumption are datetime and instant power (P(W)). These fields are loaded in Pandas dataframes in this stage.

```
|datetime|timestamp|I(amps)|P(W)|F(mm/dd/aaaa)|H(hh/mm/ss)
|Device Load (amps)|Bank 1 Load (amps)|Bank 2 Load (amps)|P(W/VA)
|Power factor|Peak Load(W)|Date Peak Load|Time Peak Load|E(KWh)
|Date since E|Time since E|Voltage(V)|Frecuency(Hz)|

|2016-08-30 00:00:00|1472526000.47|0.84|182|08/30/2016|02:54:37
|0.84|0.84|0.00|182/192
|0.947|1.82|02/07/2016|16:08:39|1120.5
|12/16/2015|12:08:30|229.0|50.0|

|2016-08-30 00:00:01|1472526001.65|0.85|183|08/30/2016|02:54:38
|0.85|0.85|0.00|183/195
|0.938|1.82|02/07/2016|16:08:39|1120.5
|12/16/2015|12:08:30|229.0|50.0|
```

Listing 4.1: Sample lines on the energy log file

Table 4.2 shows the structure of the power consumption dataframe and an example of its content.

datetime	P(W)
2016-09-08 00:00:00	186
2016-09-08 00:00:01	184
2016-09-08 00:00:02	184
2016-09-08 00:00:03	184
2016-09-08 00:00:05	184

Table 4.2: Example of content of PC dataframes

Reading start and end of benchmark executions. When a benchmark execution begins or ends, an empty text file is created. The file name contains the following information:

- Experiment id: is a code that summarizes the benchmark used (CPU, memory, or disk), type of experiment (single or combined), and the utilization level.
- Execution id: is an identifier that allows relating a file with one particular benchmark execution.
- Datetime: is the field that stores the timestamp of the file creation, i.e., the time when a benchmark execution starts or ends.

- Start or end flag: this flag indicates whether the text file corresponds to the beginning or end of the execution. The possible values of the flag are *ini* or *end*.

Listing 4.2 shows an example of file names of benchmark executions. The characters underscore (_) and period (.) are used for separating the fields described above.

```
c1f1l2525_20160912235624996_2016|09|12-23:56:25:001.ini
c1f1l2525_20160912235624996_2016|09|12-23:57:39:230.end
c1l12y5_20170227195601835_2017|02|27-19:56:01:841.ini
c1l12y5_20170227195601835_2017|02|27-19:57:02:197.end
c1l25_20170227205014281_2017|02|27-20:50:14:287.ini
```

Listing 4.2: Sample file names of benchmark executions

In this stage, the file names are loaded in dataframes and merged according to their execution id. This way, each row of the dataframes corresponds to one benchmark execution and it contains the timestamp of the begin and the end of the execution. Table 4.3 shows the structure of the power consumption dataframe and an example of its content. Column *exp* is the experiment id. The example corresponds to the first five rows of single execution on Intel host.

exp	execution id	start datetime	end datetime
c1l100	..8816	2017-02-27 21:57:18	2017-02-27 21:58:21
m2l87y5	..1724	2017-02-28 17:27:21	2017-02-28 17:28:20
m2l12y5	..0025	2017-02-28 14:01:30	2017-02-28 14:01:56
c1l25	..7249	2017-02-27 18:00:07	2017-02-27 18:01:07
m2l100	..0127	2017-02-28 18:24:00	2017-02-28 18:25:06

Table 4.3: Example of content of benchmark execution dataframes, in the reading stage

Pre-processing data. Before combining the dataframes that contains the benchmark executions information with the dataframes that contains the power consumption information, it is necessary to execute the following operations over the benchmark executions dataframes, to facilitate the visualization and the analysis. These operations include:

- Splitting the experiment code into benchmark cod and utilization level.
- Adding the offset of seconds between host and logging machine.

- Assigning variables with idle power consumption.

The difference of the system time between the host and the logging machine is calculated at the beginning of each experiment. This way, data is adjusted with the corresponding offset. Table 4.4 shows an example of dataframe containing the output of pre-processing stage. Column *exp* is the experiments code and column *bk* is the benchmark code.

exp	bk	UL	execution id	start datetime	end datetime
c1l100	c1	100	..8816	17-02-27 21:57:18	17-02-27 21:58:21
m2l87y5	m2	87.5	..1724	17-02-28 17:27:21	17-02-28 17:28:20
m2l12y5	m2	12.5	..0025	17-02-28 14:01:30	17-02-28 14:01:56
c1l25	c1	25	..7249	17-02-27 18:00:07	17-02-27 18:01:07
m2l100	m2	100	..0127	17-02-28 18:24:00	17-02-28 18:25:06

Table 4.4: Example of content of benchmark execution dataframes, in the pre-proccesing stage

The variables *AMD_IDLE_PC* and *INTEL_IDLE_PC*, which store the idle power consumption of each host respectively, are assigned at this stage for use in the calculation of effective energy consumption.

Crossing data sources. After collecting data from each source and performing pre-processing operations, it is necessary to cross the information in order to obtain the power consumption of each benchmark executions. For each experiment id, that corresponds to the combination benchmark, the type of experiment (i.e. single or combined), and the utilization level, 20 independent executions are performed. The result of this stage are dataframes grouped by the type of experiment and host, where thier rows correspond to benchmark executions and their power consumption, calculated as the average of the power registered in the logging machine while the benchmark is executing. Table 4.5 presents an example of Pandas dataframe with the power consumption information of benchmark executions on one host. Column *exp* is the experiments code, column *bk* is the benchmark code, and column *exe* is execution id. These particular example corresponds to the dataframe that contains the data of single experiments and Intel host.

Show results. At the end of the processing, the results of benchmark executions are showed grouped by benchmark code, UL and host, and applying

exp	bk	UL	exe	start datetime.	end datetime.	P(W)
c1l100	c1	100	..8816	17-02-27 21:57:18	17-02-27 21:58:21	194.3
m2l87y5	m2	87.5	..1724	17-02-28 17:27:21	17-02-28 17:28:20	242.1
m2l12y5	m2	12.5	..0025	17-02-28 14:01:30	17-02-28 14:01:56	132.7
c1l25	c1	25	..7249	17-02-27 18:00:07	17-02-27 18:01:07	134.7
m2l100	m2	100	..0127	17-02-28 18:24:00	17-02-28 18:25:06	235.5

Table 4.5: Example of dataframe containing the PC of benchmark executions, in the crossing stage

average operation of *P(W)* column, to obtain the mean value of the independent executions. Table 4.6 presents an example of the power consumption of single experiments of CPU, memory and disk intensive benchmarks, in both AMD and Intel hosts.

UL	AMD CPU	AMD mem	AMD dk	Intel CPU	Intel mem.	Intel dk
12.5	194.4	215.5	188.1	121.6	125.9	67.5
25	205.0	238.7	189.3	136.6	156.3	68.0
37.5	215.1	257.7	189.5	142.5	178.7	67.7
50	225.1	272.3	189.3	152.8	191.8	68.9
62.5	235.5	278.5	189.9	163.2	213.0	70.5

Table 4.6: Example of dataframe containing PC of benchmark executions, in the results stage.

Performance data extraction procedure Data of performance experiments also are processed using *Pandas*. In this case, there is only one data source. Figure 4.7 presents the procedure data processing pipeline for performance experiments.

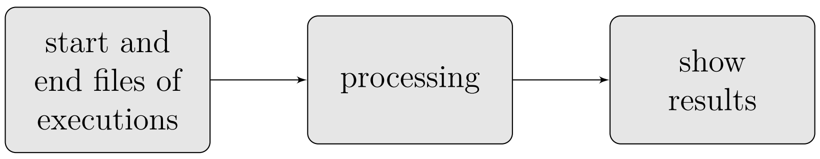


Figure 4.7: Procedure for performance data extraction

In performance experiments, the same benchmarks without time limit are executed, at different utilization levels. This procedure allows determining how the load affects the performance for each type of benchmark. Only single experiments are considered.

Start and end files of executions. When a benchmark execution begins or ends, a text file is created, containing the following information in its name:

- Experiment id: an identifier that resumes the benchmark utilized (CPU, memory o disk), the modality (single or combined) and the utilization level.
- Execution id: An identification of the particular execution. An execution consists of several benchmark instances and an instance is executed in only one core. The number of benchmark instances depends on the corresponding utilization level.
- Instance id: an identification of a benchmark instance in the execution.
- Datetime: the timestamp of the file creation.
- Begin or end flag: a flag that indicates whether is the beginning or the end of the execution.

An example of the format of file names of benchmark instance executions is showed in Listing 4.3. The characters underscore (_) and period (.) are used for separating the fields described above.

```
c1ntl12y5_20170302122631609_20170302122631614 \
_2017|03|02 -12:26:31:620.ini
f1ntl75_20170207025026035_20170207025027159 \
_2017|02|07 -03:24:19:119.end
m2ntl50_20170131043902686_20170131043903301 \
_2017|01|31 -04:39:03:306.ini
f1ntl87y5_20170207194619767_20170207194621197 \
_2017|02|07 -20:24:19:705.end
c1ntl87y5_20170130030621714_20170130030621819 \
_2017|01|30 -03:07:06:187.end
```

Listing 4.3: Sample file names of benchmark instance executions

In this stage, the file names are loaded in the dataframes and merged according to the instance id. Each row of the dataframes corresponds to one benchmark instance and it contains the timestamp of the begin and the end of the execution. In addition, the duration of the instance execution is inferred from the timestamps and added as a new column of the dataframe. Table 4.7 shows an example of the dataframe content. Column *exp* is the experiments code, column *bk* is the benchmark code, column *exe* is execution id, column *ins* in the instance id, column *start dt* is the start datetime, and column *end* *dt* is the end datetime. The example shows the first five rows of the dataframe that corresponds to AMD host.

exp	bk	UL	exe	ins	start dt	end dt	duration
f1ntl75	f1	75.0	..6035	..7159	..02:50:27	..03:24:19	2032
m2ntl50	m2	50.0	..2686	..3301	..04:39:03	..04:41:54	171
f1ntl87y5	f1	87.5	..9767	..1197	..19:46:21	..20:24:19	2278
c1ntl87y5	c1	87.5	..1714	..1819	..03:06:21	..03:07:06	45
c1ntl100	c1	100	..5919	..8152	..03:14:08	..03:14:52	44

Table 4.7: Example of dataframe containing performance of benchmark executions, in reading stage

Processing. In this stage, the performance execution data is processed by applying the following operations on the dataframe, to obtain the metric of makespan: The rows are grouped by benchmark, UL, host and execution id, applying the maximum function as the grouping criterion. In this way, the value obtained in the duration column is the makespan of the execution. Later, the rows of the resulting dataframe are grouped by benchmark, UL and host, with the average function as the grouping criterion. The final result of the duration column is the average makespan of the experiments for each host, benchmark and UL. In the final dataframe the duration column is renamed as makespan.

Show results of performance experiments At the end of the processing, the results of benchmark executions are reported. Table 4.8 presents the five first row of the result dataframe that contains the makespan of CPU, memory and disk intensive benchmarks on both host at all utilization levels.

UL	AMD CPU	AMD mem	AMD dk	Intel CPU	Intel mem	Intel dk
12.5	44.4	90.2	234.7	35.8	34.6	36.0
25	44.9	136.8	547.7	30.4	26.6	74.0
37.5	45.0	179.8	839.4	36.2	41.1	114.7
50	45.0	220.8	1397.8	30.8	41.8	154.5
62.5	45.0	259.1	1644.0	29.5	67.5	195.9

Table 4.8: Example of dataframe containing performance of benchmark executions, in the results stage

4.3.3 Building models using Python libraries for analysis of scientific data

Models were built using scientific Python libraries for supervised machine learning techniques. The input of modeling step is the output of data extraction step, described above. The applied methodology allows an easy and fast error correction. Also, models can be adjusted varying parameters or adding new experimental information. Through Jupyter Notebook tool, published at https://www.fing.edu.uy/~jmurana/msc, is able to obtain an ordered documentation of the process.

The models were implemented using statsmodels (Seabold and Perktold, 2010), a library for statistical and econometric analysis. The library allows indicating methods (OLS, IRLS, etc.), variable dependencies and domain. The output of the library is a function that adjusts a set of values, considering the specified parameters. In this research, models functions were generated using ordinary least squared (OLS) and linear, quadratic an cubic dependency. This configuration were used for models with one, two and three independent variables as function domain.

Listing 4.4 shows a snipped of Python code used for modeling construction, where energy_data is a dataframe with columns UL and power. These columns corresponds to the previous processing described in Subsection 4.3.2. Parameter formula corresponds to the dependency written in R language style (R Core Team, 2018) (in this example, linear dependency between domain and co-domain is indicated).

```
import pandas as pd
import statsmodels.formula.api as smf

model = smf.ols(formula='power␣~␣UL', data=energy_data).fit()
```

Listing 4.4: Model construction example using *statsmodel* library

In Listing 4.5, the model is used for predicting values in the considered domain, i.e., the interval [0,100]). Later, the obtained values are plotted. The output of plottings are presented and analyzed for each case in Chapter 6.

```
import matplotlib.pyplot as plt

domain = pd.DataFrame({'UL': [0, 100]})
co_domain = model.predict(domain)
plt.plot(domain, co_domain)
```

Listing 4.5: statsmodel usage example

The modeling library provides the method *summary()*, which allows to easily obtain model statistics for validating and comparison. The usage and output of *summary()* is showed in Listing 4.6.

```
print(model.summary())

OLS Regression Results
==============================================================================
Dep. Variable:                  power   R-squared: 0.999
Model:                            OLS   Adj. R-squared: 0.999
Method:                 Least Squares   F-statistic: 9523.
Date:                Wed, 09 Jan 2019   Prob (F-statistic): 7.80e-11
Time:                        09:31:29   Log-Likelihood: -6.9237
No. Observations:                   8   AIC: 17.85
Df Residuals:                       6   BIC: 18.01
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
           coef      std err   t        P>|t|   [0.025      0.975]
------------------------------------------------------------------------------
Intercept  184.9822  0.517     357.593  0.000   183.716     186.248
UL         0.7998    0.008     97.588   0.000   0.780       0.820
==============================================================================
Omnibus:                        0.257   Durbin-Watson: 1.129
Prob(Omnibus):                  0.879   Jarque-Bera (JB): 0.211
Skew:                           0.258   Prob(JB): 0.900
Kurtosis:                       2.394   Cond. No. 139.
==============================================================================
```

Listing 4.6: *statsmodel* library stats usage and output example

In performance models, the modeling procedure is similar to energy ones. In this case, the dependent variable is the makespan instead of the power consumption. Listing 4.7 shows an example of code of cubic dependency between UL and makespan.

```
import numpy as np

model = smf.ols(formula='makespan ~ UL + np.power(UL,2) + \
np.power(UL,3)', data=performance_data).fit()
```

Listing 4.7: Performance model construction using *statsmodel* library

Chapter 5

Power and performance evaluation results

This chapter reports and discuses the results of power consumption and performance evaluation for AMD and Intel architectures. Section 5.1 presents tables and graphics of results of benchmark executions, which are executed single and combined in order to obtain power measurement. Also, the observed behavior of each execution is discussed. Section 5.2 presents and discusses the results of performance evaluation. Finally, Section 5.3 presents a efficiency study, that combines results of power consumption and performance evaluation.

5.1 Results of executions

This section presents the results obtained in experiments that studied the power consumption. Details of idle power consumption evaluation are presented in Subsection 5.1.1. Subsection 5.1.2 shows and discusses the results of the power consumption evaluation considering benchmarks execution independently. Then, the results of combined benchmark executions are presented in Subsection 5.1.3.

5.1.1 Idle power consumption evaluation

The average idle power consumption in both AMD and Intel hosts was calculated by performing 20 independent executions of a null program, i.e, a program that executes a sleep function for 60 seconds. The number of independent executions was chosen to obtain results with statistical validity. The average values obtained for idle power consumption (± standard deviation) were 183.4±1.3 W for the AMD host and 57.0±0.9 Watts for the Intel host. These average values are considered as the idle power consumption, of the corresponding host, in all the experiments reported in this section. For example, if the measured overall consumption of one experiment in Intel host was 200 Watts, the effective power consumption of the experiment was calculated as 200 Watts less 57 Watts, this is, 143 Watts.

5.1.2 Results of single benchmark executions

CPU-bound benchmark. Table 5.1 reports PC and EC values for the CPU-bound benchmark for Intel and AMD hosts. The PC is reported as the average more less the standard deviation of the independent executions of the experiment. The PC results indicate that the AMD host demands more power than the Intel host for CPU-bound workload. The PC difference between hosts is approximately 73 Watts, which signify that the AMD host consumes approximately 60% more power than the Intel host. However, it is not possible conclude about the hosts energy efficiency without a performance analysis.

According to peak PC reported by Table 5.1 (264.1 Watts on AMD and 194.6 Watts, that occurs at UL 100%), the IC represents the 69% on AMD and 29% on Intel, of the maximum power consumption.

Table 5.1: PC and EC results for the CPU-bound benchmark on AMD and Intel hosts.

AMD			*Intel*		
UL	*PC*	*EC*	*UL*	*PC*	*EC*
12.5%	194.4±0.5	11.0	12.5%	121.6±1.7	64.6
25.0%	205.0±0.7	21.6	25.0%	136.6±2.0	79.6
37.5%	215.1±0.5	31.7	37.5%	142.5±2.1	85.5
50.0%	225.1±0.8	41.7	50.0%	152.8±2.9	95.8
62.5%	235.5±1.6	52.1	62.5%	163.2±2.7	106.2
75.0%	246.0±1.4	62.6	75.0%	175.8±1.8	118.8
87.5%	254.7±1.0	71.3	87.5%	185.8±2.3	128.8
100.0%	264.1±1.8	80.7	100.0%	194.6±4.3	137.6

Figure 5.1 presents a graphic comparison of EC values in both hosts, which shows an average EC difference of 56 Watts between Intel and AMD hosts for

all ULs. However, this difference is not relevant in order to evaluate which of both host consumes more power, because the absolute value of EC is not representative of the overall power consumption of the host. In the proposed graphic analysis, the relevant indicator is the variation of EC regarding the utilization level and also the difference (of the EC variation) between hosts (i.e., a comparison of its derivatives). The almost linear behavior of EC in Figure 5.1 indicates that power consumption is proportional to the UL. Furthermore, curves for Intel and AMD are almost parallel, indicating that the power consumption of CPU-bound applications has a similar behavior in both hosts. EC on Intel host shows a remarkable increase when moving from 0 to 12.5% UL, which is not observed for the other ULs. This increase is explained by the dynamic handling of chip power, awaking, or increasing voltage of its components according to usage demand, which is notorious on Intel architecture. In critic UL (%100), it is not observed any different behavior regarding the power consumption, which allows concluding that in CPU-bound workloads, resource conflict does not implies an increase in power consumption.

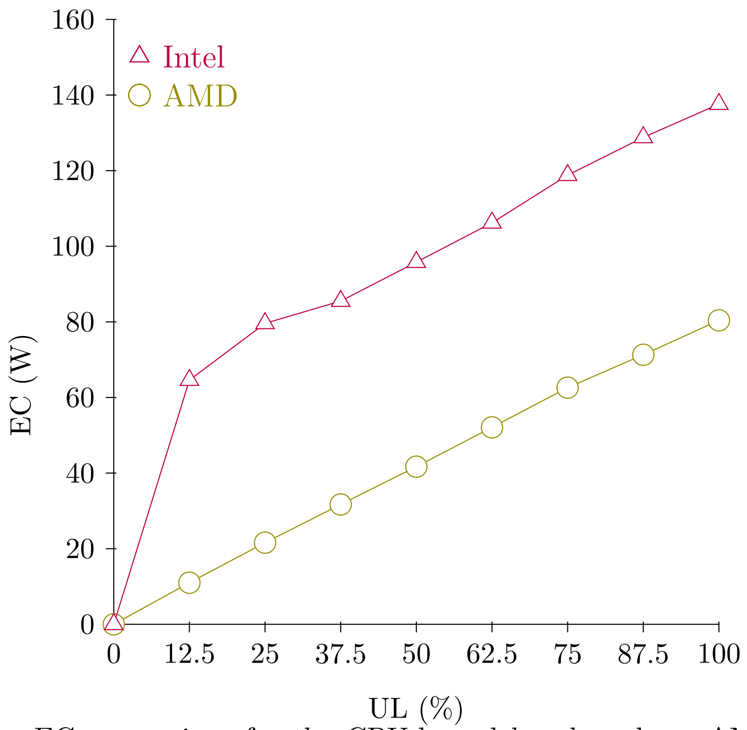


Figure 5.1: EC comparison for the CPU-bound benchmark on AMD and Intel hosts

Memory-bound benchmark. Table 5.2 reports the PC and EC values for the memory-bound benchmark for Intel and AMD hosts. The comparison between PC values of memory experiments and the CPU (presented above) allows concluding that memory use has impact in power consumption, because at same level and host, the power consumption of memory experiments is greater than CPU experiments. As seen in the CPU experiments, the AMD host consumes more power than Intel. The biggest difference in PC (89.6 Watts) occurs at the minimum UL (12%) and the lowest (45.7 Watts) occurs at 100%, indicating that Intel has an efficient management of the energy with respect to the use memory.

Table 5.2: PC and EC results for the memory-bound benchmark on AMD and Intel hosts

AMD			*Intel*		
UL	*PC*	*EC*	*UL*	*PC*	*EC*
12.5%	215.5±1.8	32.1	12.5%	125.9±6.4	68.9
25.0%	238.7±1.1	55.3	25.0%	156.3±4.8	99.3
37.5%	257.7±1.9	74.3	37.5%	178.7±4.8	121.7
50.0%	272.3±2.7	88.9	50.0%	191.8±6.8	134.8
62.5%	278.5±6.4	95.1	62.5%	213.0±3.9	156.0
75.0%	279.9±5.9	96.5	75.0%	225.1±5.4	168.1
87.5%	290.8±5.6	107.4	87.5%	239.2±4.0	182.2
100.0%	294.7±3.0	111.3	100.0%	249.0±9.3	192.0

In order to analyze the PC difference between CPU and memory bound workload, Figure 5.2 presents ΔPC, defined as the additional percentage of power that memory experiments consumes regarding CPU experiments (i.e, $\Delta PC = (PC_{MEM} - PC_{CPU}) \times 100/PC_{CPU}$). Since the CPU-bound benchmark is designed to consume more CPU cycles than the memory-bound benchmark, ΔPC is related to the power consumption of the memory usage. The graphic comparison of ΔPC allows observing a remarkable increase of ΔPC on Intel at medium and high ULs, when compared with low UL of the same host. On AMD, there is not a notorious ΔPC increment. This difference between hosts suggests that the power management on Intel architecture is better than on AMD architecture, since the power consumption increases according to the percentage of used memory (the percentage of the system memory used by the benchmark is equal to the UL, for example, 50% of the host total memory is used at UL 50%).

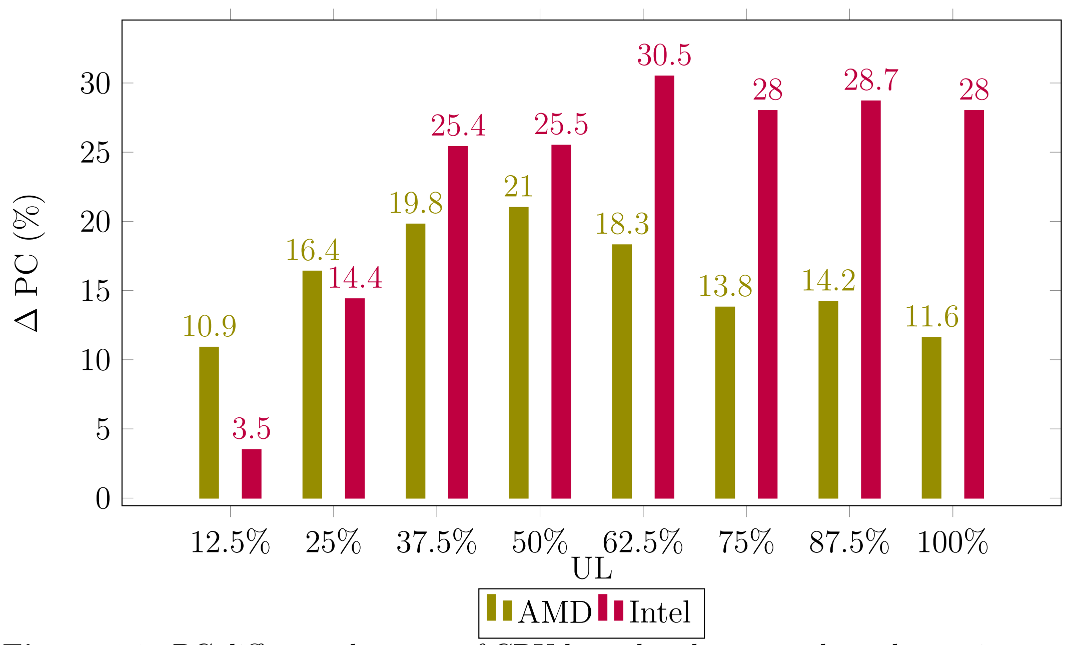


Figure 5.2: PC difference between of CPU-bound and memory-bound experiments

Figure 5.3 presents a graphic comparison of the EC values in both hosts. Results show a significant increase of EC with regard to CPU-bound executions for all ULs (104% for the AMD host and 36% for the Intel host, on average).

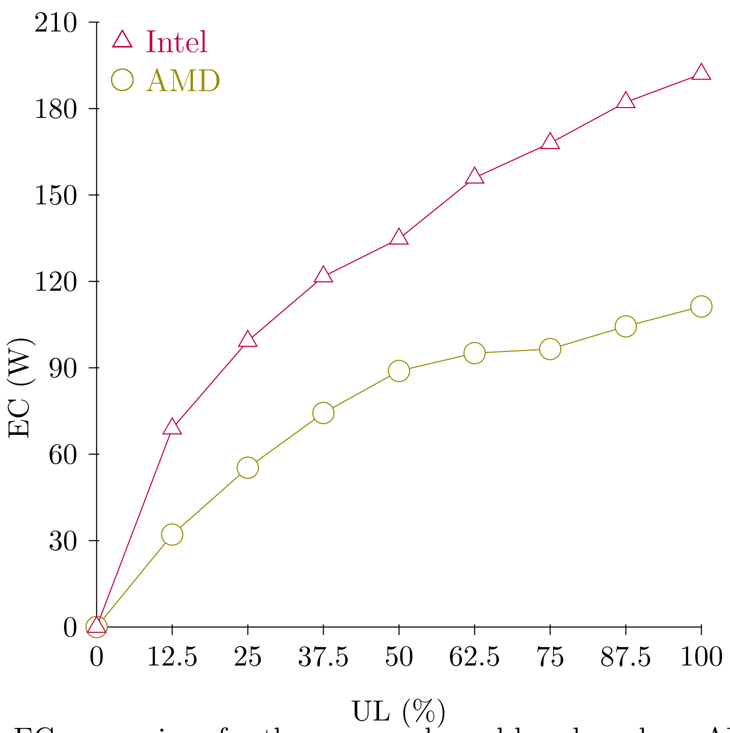


Figure 5.3: EC comparison for the memory-bound benchmark on AMD and Intel hosts

A logarithmic behavior is observed for both PC and EC, which does not occur in CPU-bound case. This behavior may be mainly due to the bottleneck in the access to the main memory that reduces the CPU usage. No significant increase is detected on high/critical ULs, possibly by effective resource contention by the operating system for solving conflicts over access to shared resources.

Disk-bound benchmark. Table 5.3 reports PC and EC values for the disk-bound benchmark. The PC is almost constant and notoriously less than CPU and memory experiments.

Table 5.3: PC and EC results for the disk-bound benchmark on AMD and Intel hosts

AMD			*Intel*		
UL	*PC*	*EC*	*UL*	*PC*	*EC*
12.5%	188.1±0.4	4.7	12.5%	67.5±0.9	10.5
25.0%	189.3±0.5	5.9	25.0%	68.0±0.6	11.0
37.5%	189.5±0.6	6.1	37.5%	67.7±0.6	10.7
50.0%	189.3±0.3	5.9	50.0%	68.9±0.7	11.9
62.5%	189.3±0.3	6.5	62.5%	70.5±0.7	13.5
75.0%	190.1±0.4	6.7	75.0%	70.7±0.8	13.7
87.5%	190.7±1.6	7.3	87.5%	71.7±0.6	14.7
100.0%	190.4±0.6	7.0	100.0%	72.4±1.0	15.4

Figure 5.4 presents a comparison of EC values in both hosts. The maximum EC variation through ULs is 4W in Intel and 2W in AMD. These low power variation indicate that disk usage has low impact in power consumption in comparison with CPU and memory.

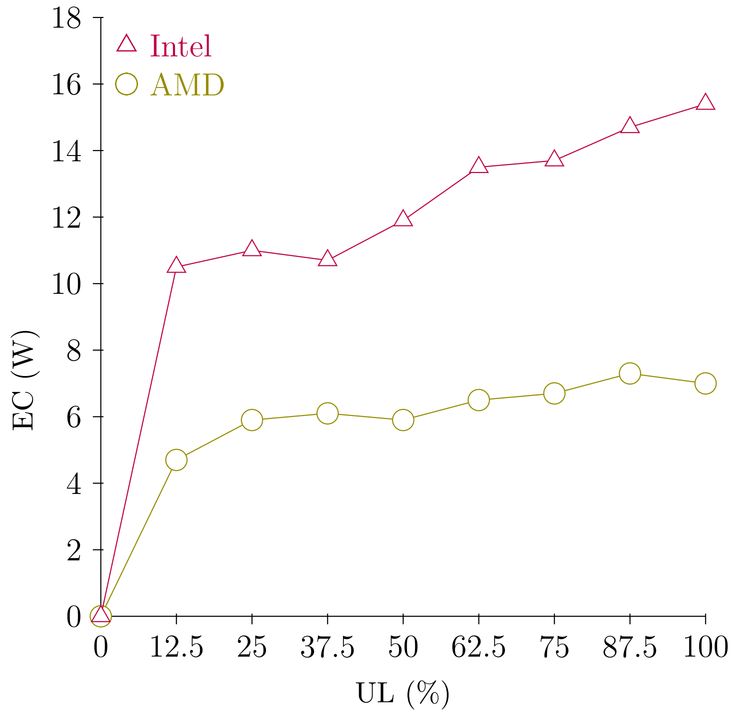


Figure 5.4: EC comparison for the disk-bound benchmark on AMD and Intel hosts

5.1.3 Results of combined benchmark executions

CPU- and memory-bound benchmarks. Table 5.4 reports PC and EC for the simultaneous execution of the CPU- and memory-bound benchmarks on AMD host. The results shows that the peak PC on AMD is 312.7 Watts and it occurs at UL (25%,75%). This peak is the maximum PC on AMD host registered considering all experiments in this research, indicating that the IC on AMD (183.4 Watts) represents 59% of the total PC of the host.

Table 5.4: PC and EC results for the CPU- and memory-bound benchmarks combined on AMD host

UL (CPU,memory)	*PC*	*EC*	ΔEC
(25%,25%)	259.4±2.7	76.0 =	0.77
(25%,50%)	290.3±1.6	106.9 ↓	3.59
(25%,75%)	312.7±3.3	129.3 ↑	-11.30
(50%,25%)	277.1±2.2	93.7 ↓	3.21
(50%,50%)	310.3±2.6	126.9 ↓	3.69
(75%,25%)	295.8±6.6	112.4 ↓	5.53

Symbol ↑ indicates that the EC of the combined benchmarks is higher than the sum of the ECs of each benchmark executed independently, i.e., the combined execution is less efficient than the independent execution. Symbol ↓ indicates the opposite, that is, the combined execution is more efficient. Symbol = indicates that the values are equal, considering a threshold of 1 W of difference. Column ΔEC reports the difference between the EC of the combined execution and the sum of the EC of the independent executions (in Watts).

The ΔEC values reported in Table 5.4 shows that the combined execution on AMD host allows reduce the EC compared to independent executions. In UL with high memory use, the independent executions presents lower EC values that combined.

The 3D graph on Figure 5.5 presents the EC values of the combined execution of CPU and memory on AMD host. The color scale of the graph allows observing that the dark sectors, which correspond to a high EC, are accentuated near the memory axis.

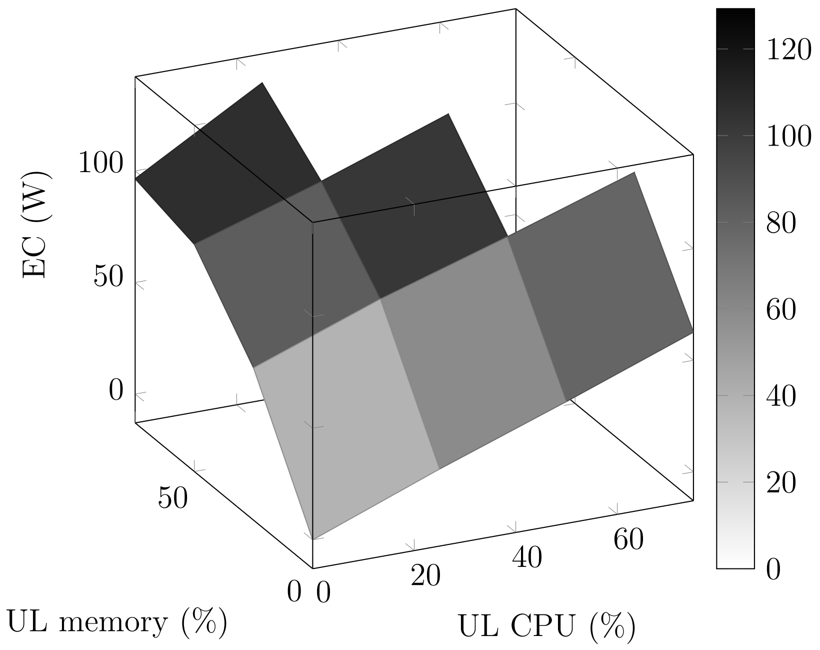


Figure 5.5: Combined CPU- and memory-bound EC on AMD host

Table 5.5 reports PC and EC for the simultaneous execution of the CPU- and memory-bound benchmarks on Intel host. The peak PC (255.3 Watts) corresponds to UL (50%,50%), thus, the IC of the Intel host represents the 22% of the overall PC. The low percentage of IC is due to Intel host reduces the IC when the host is not used (at UL almost 0%), however, in other ULs the IC increases (as is observed and discussed in Section 5.1, a significant increment of PC occurs in the first UL, which does not occurs in later ULs). The combined executions of CPU and memory benchmarks in Intel host achieve the best results with regard to EC in pair combined experiments. In this case, a significant gain (around 47 Watts) is observed for all ULs. The highest gain occurs when CPU use is low and memory use is low, simultaneously.

Table 5.5: PC and EC results for the CPU- and memory-bound benchmarks combined on Intel host

UL (CPU,memory)	*PC*	*EC*	ΔEC
(25%,25%)	178.2±5.0	121.2 ⇃	57.65
(25%,50%)	226.2±3.6	169.2 ⇃	45.10
(25%,75%)	254.5±5.6	197.5 ⇃	50.08
(50%,25%)	199.9±3.5	142.9 ⇃	52.23
(50%,50%)	255.3±3.2	198.3 ⇃	32.28
(75%,25%)	225.0±3.2	168.0 ⇃	50.09

The 3D graph on Figure 5.6 presents the EC values of the combined execution of CPU and memory on AMD host. The graph allows observing the difference in derivative of UL 0 and the other ULs and the dark sectors near memory axis.

Results of combined CPU and memory benchmarks show that for Intel host, the combined executions reduce EC compared to independent executions. In AMD host, however, the combined executions of these types of benchmarks have not such notorious improvement when compared with single executions.

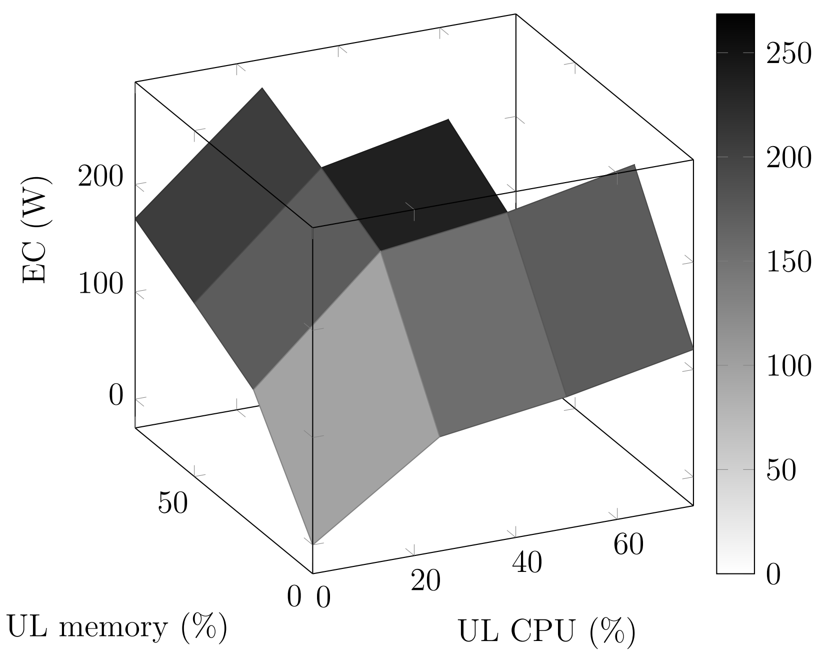


Figure 5.6: Combined CPU- and memory-bound EC on Intel host

CPU- and disk-bound benchmarks. Table 5.6 reports PC and EC values for the simultaneous execution of the CPU- and disk-bound benchmarks on AMD host. ΔEC values indicate that the combined execution has not a significant gain or loss regarding independent execution.

Table 5.6: PC and EC results for the CPU- and disk-bound benchmarks combined on AMD host

UL (CPU,disk)	*PC*	*EC*	*ΔEC*
(25%,25%)	210.0±1.1	26.6 =	0.83
(25%,50%)	211.5±1.3	28.1 =	-0.67
(25%,75%)	212.4±1.2	29.0 =	-0.71
(50%,25%)	229.8±2.6	46.4 ↓	1.14
(50%,50%)	234.3±4.9	50.9 ↑	-3.38
(75%,25%)	248.4±3.5	65.0 ↓	3.49

The 3D graph on Figure 5.7 presents EC values on AMD host, for different combinations of CPU and disk ULs. The graph shows that EC values only change significantly in the direction of the CPU axis, which indicates that the disk-bound load has not significant impact on EC.

Table 5.7 reports PC and EC values for the simultaneous execution of the

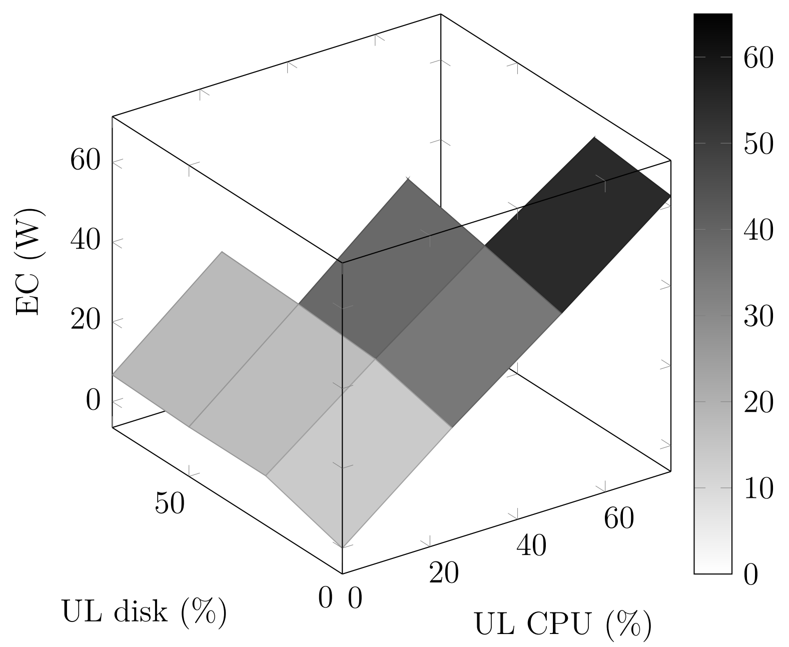


Figure 5.7: Combined CPU- and disk-bound EC on AMD host

CPU- and disk-bound benchmarks on Intel host. Results show an average gain of 10 Watts of combination regarding to independent execution. The maximum gain is 13.06 Watts and it occurs at UL (25%,75%).

Table 5.7: PC and EC results for the CPU- and disk-bound benchmarks combined on Intel host

UL (CPU,disk)	*PC*	*EC*	ΔEC
(25%,25%)	136.4±2.3	79.4 ↓	11.19
(25%,50%)	139.0±1.9	82.0 ↓	9.40
(25%,75%)	137.2±3.2	80.2 ↓	13.06
(50%,25%)	154.2±2.6	97.2 ↓	9.64
(50%,50%)	156.5±2.8	99.5 ↓	8.19
(75%,25%)	177.8±2.9	120.8 ↓	8.96

The 3D graph on Figure 5.8 presents EC values on Intel host, for different combinations of CPU and disk ULs. A similar result to the one obtained in AMD is observed, as EC changes significantly only in CPU axis direction. The aforementioned remarkable EC increase when moving from 0 to the next UL is notorious on CPU axis direction, but not on disk axis direction. This results indicate that the increment is related to CPU use.

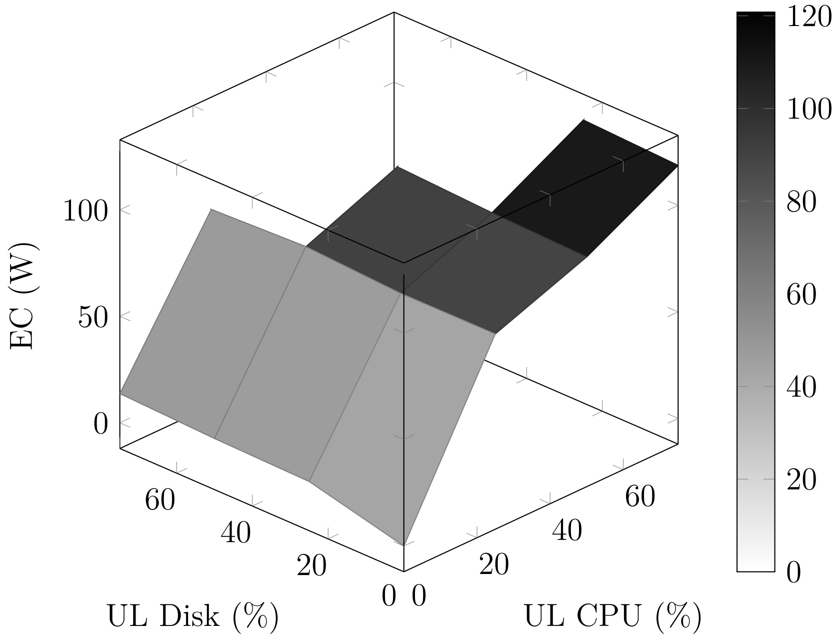


Figure 5.8: Combined CPU- and disk-bound EC on Intel host

Memory- and disk-bound benchmarks. Table 5.8 reports PC and EC values for the simultaneous execution of the memory- and disk-bound benchmarks on AMD host. The combined executions consume less energy at ULs (50%,25%) and (50%,50%), i.e., when memory-bound load is 50%. In the other ULs, there are not significant improvements on power consumption.

Table 5.8: PC and EC results for the memory- and disk-bound benchmarks combined on AMD host

UL (memory,disk)	*PC*	*EC*	ΔEC
(25%,25%)	243.9±1.5	60.05 =	0.61
(25%,50%)	241.8±5.5	58.4 ↓	2.81
(25%,75%)	245.9±2.0	62.5 =	-0.54
(50%,25%)	268.2±9.1	84.8 ↓	10.01
(50%,50%)	269.5±12.9	86.1 ↓	8.71
(75%,25%)	287.0±4.0	103.6 ↑	-1.19

The 3D graph on Figure 5.9 presents EC values on AMD host for memory and disk-bound benchmarks executing in combination. The graph allows observing the same behaviour noted in the analysis of Figure 5.8 regarding the low increment of EC in disk axis direction, which confirms the almost negligible power consumption of disk-bound benchmark.

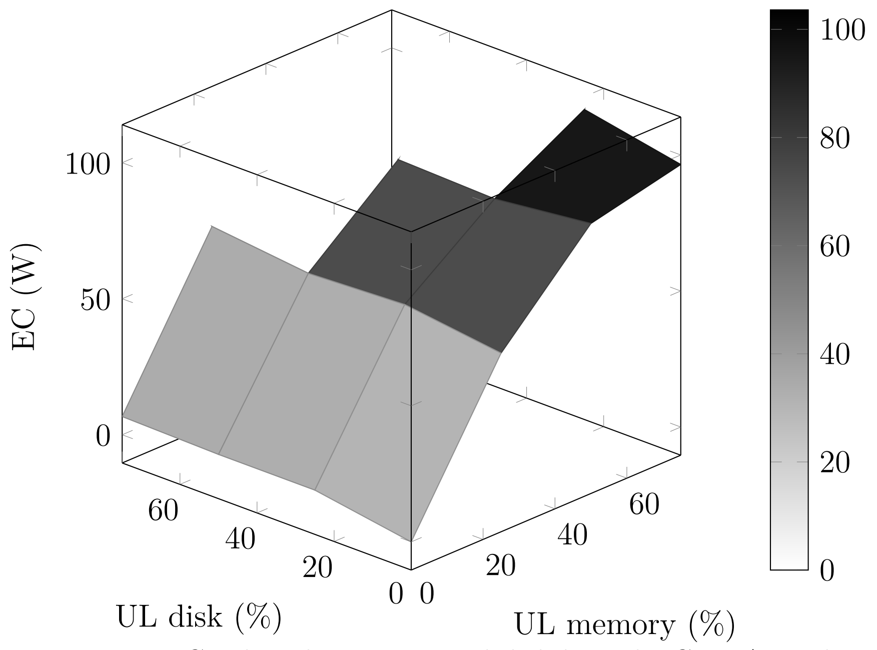


Figure 5.9: Combined memory- and disk-bound EC on AMD host

Table 5.9 reports PC and EC values for the simultaneous execution of the memory- and disk-bound benchmarks on Intel host. The results show that single executions consume less EC than combined, except for those experiments with high load of the memory-bound benchmark.

Table 5.9: PC and EC results for the memory- and disk-bound benchmarks combined on Intel host

UL (memory,disk)	*PC*	*EC*	ΔEC
(25%,25%)	174.1±8.3	117.1 ↑	-6.70
(25%,50%)	176.2±5.9	119.2 ↑	-8.01
(25%,75%)	172.0±7.6	115.0 ↑	-2.00
(50%,25%)	206.4±6.0	149.4 ↑	-3.66
(50%,50%)	208.6±4.2	151.6 ↑	-4.93
(75%,25%)	228.5±5.0	171.5 ↓	7.51

The 3D graph on Figure 5.10 presents EC values on Intel host for memory and disk-bound benchmarks executing in combination. A similar results to the one obtained in AMD is observed, where the disk-bound benchmark has not impact in EC.

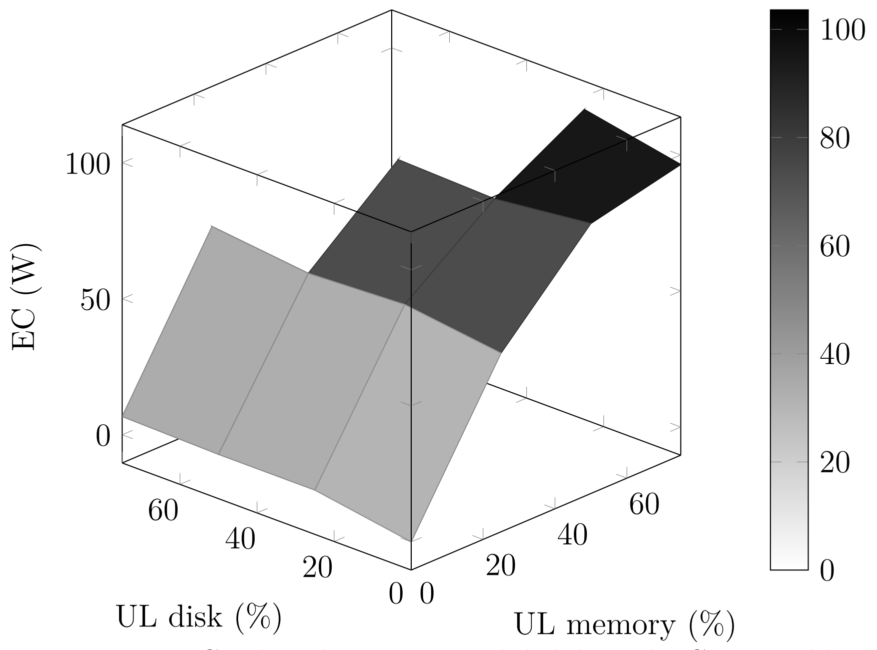


Figure 5.10: Combined memory- and disk-bound EC on Intel host

CPU-, memory-, and disk-bound benchmarks. Table 5.10 reports the PC and EC values obtained when executing the CPU-, memory- and disk-bound benchmarks combined, on AMD and Intel hosts.

Table 5.10: PC and EC results for the CPU-, memory- and disk-bound benchmarks combined

UL	*AMD*		*Intel*	
(CPU,mem.,disk)	*PC*	*EC*	*PC*	*EC*
(25%,25%,25%)	266.3±4.5	82.9 =	177.4±4.1	120.4 ↓
(25%,25%,50%)	266.9±4.7	83.5 =	178.7±3.5	121.7 ↓
(25%,50%,25%)	303.2±3.3	119.8 ↑	220.9±5.5	163.9 ↓
(50%,25%,25%)	288.1±1.7	104.7 ↑	195.0±5.0	138.0 ↓

Results in Table 5.10 show that the combined execution on the AMD host has a higher EC compared to their independent execution in ULs (25%,50%,25%) and (50%,25%,25%). This behavior indicates that the three-combined execution is not efficient for high load of CPU-bound and memory-bound benchmarks. However, on Intel host, combined executions reduce EC compared to independent executions for all ULs. For independent executions on Intel host, the difference on PC between consecutive ULs is larger at the first UL (12.5%). Because of this difference, combined executions achieve better efficiency regarding independent execution on Intel host.

.

5.2 Performance evaluation

This section presents a performance evaluation which complements the power consumption study, providing insight into how ULs affect performance. Both studies together (power consumption and performance degradation) allow finding the optimum UL regarding to energy efficiency for each host and type of benchmark. The details of the performance experiments design were presented in Section 4.2.

Table 5.11 reports the makespan of the CPU-bound benchmark for AMD and Intel host. Since the makespan does not present significant variations, increasing the UL of CPU-bound benchmark does not impact the completion time, which is due to the absence of resource competition. However, both hosts present a slight degradation for UL 100%, possibly due to conflicts with operating system processes.

Table 5.11: Makespan results for the CPU-bound benchmark on AMD and Intel hosts

AMD		*Intel*	
UL	*makespan*	*UL*	*makespan*
12.5%	44.4±0.5	12.5%	35.8±8.6
25.0%	44.9±0.4	25.0%	30.4±5.7
37.5%	45.0±0.0	37.5%	36.2±9.7
50.0%	45.0±0.0	50.0%	30.8±3.7
62.5%	45.0±0.0	62.5%	29.5±0.6
75.0%	45.0±0.0	75.0%	29.9±0.8
87.5%	45.0±0.0	87.5%	34.0±9.4
100.0%	50.5±4.5	100.0%	47.4±11.0

Figure 5.11 presents a graphic comparison of EC values for both hosts. The graph shows that the AMD host presents less variation of makespan through UL than Intel host. Since the experiments consist of executing the same benchmark with the same computational effort, it is possible to conclude that Intel host has a better performance than AMD host with respect to the CPU-bound workload. Considering the average makespan for all ULs, the Intel host is roughly 25% faster than the AMD host.

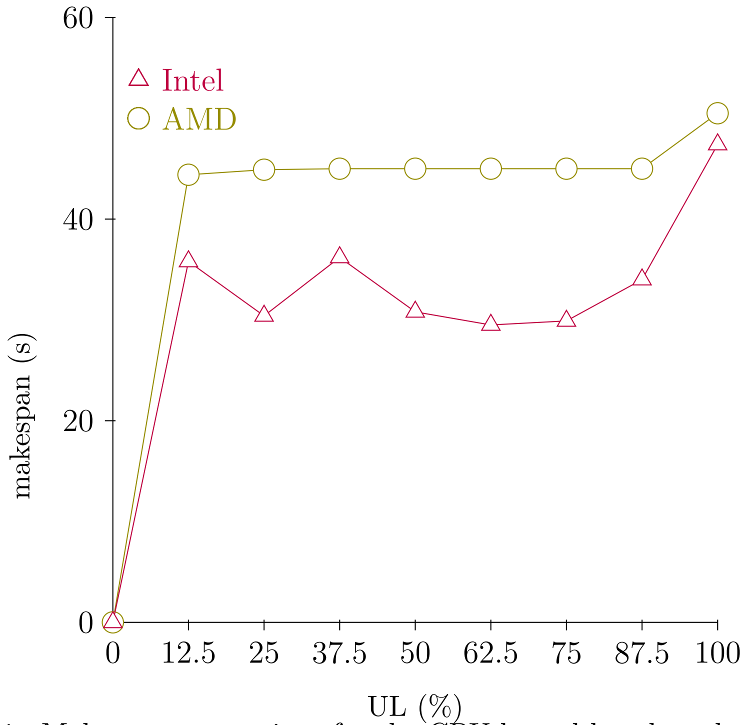


Figure 5.11: Makespan comparison for the CPU-bound benchmark on AMD and Intel hosts

Table 5.12 reports the makespan of the memory-bound benchmark for both AMD and Intel host.

Table 5.12: Makespan results for the memory-bound benchmark on AMD and Intel hosts

AMD		*Intel*	
UL	*makespan*	*UL*	*makespan*
12.5%	90.2±4.7	12.5%	34.6±4.7
25.0%	136.8±7.4	25.0%	26.6±0.9
37.5%	179.8±14.2	37.5%	41.1±2.2
50.0%	220.8±12.3	50.0%	41.8±1.1
62.5%	259.1±12.9	62.5%	67.5±4.3
75.0%	326.7±11.4	75.0%	62.2±1.7
87.5%	408.2±14.6	87.5%	74.5±1.5
100.0%	490.6±16.2	100.0%	82.5±2.0

Results in Table 5.12 demonstrate that performance degrades in AMD. There is a gap of 400 seconds between the lowest and the highest UL. For Intel the difference is only 48 seconds. The difference in gaps is possibly explained by the specific memory features of each host, such as cache size and transfer speed: the Intel host processor has a significantly larger L3 cache size than the AMD host processor (20MB vs 12MB) and a faster bus clock speed (4.800 MHz vs 3.200 MHz).

Figure 5.12 presents a graphic comparison of makespan for both hosts. The graph shows an increment of derivative on AMD host from UL 62.5 %, indicating the increasing of performance degradation at highest ULs.

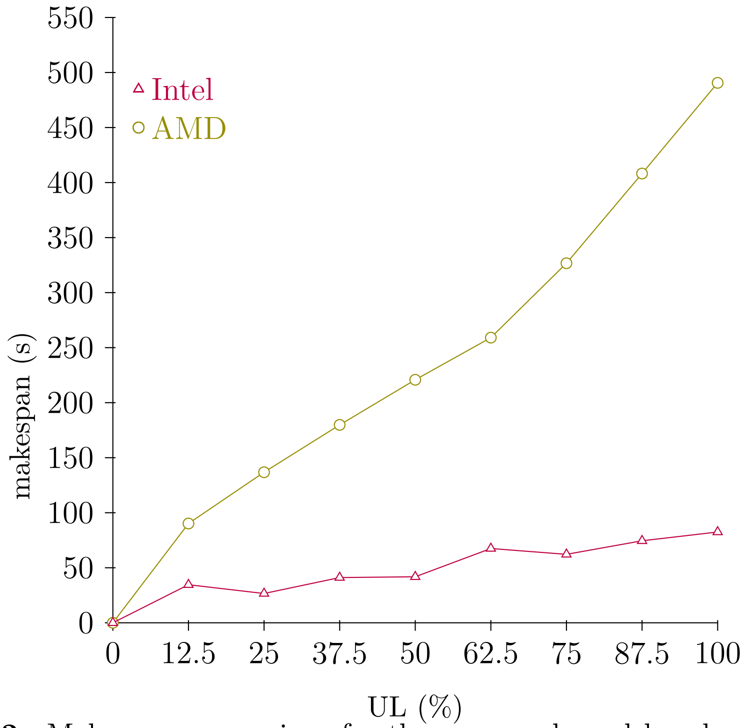


Figure 5.12: Makespan comparison for the memory-bound benchmark on AMD and Intel hosts

Table 5.13 reports the makespan of the disk-bound benchmark for AMD and Intel hosts at different ULs. The disk-bound case presents a significant degradation in performance when increasing UL when compared with other benchmarks, because concurrency has a bigger impact in the disk access than memory access and CPU access. On AMD host, the makespan of the maximum UL (100.0%) is approximately 11 times the makespan of the first UL (12.5%), and on Intel the makespan of the maximum UL (100.0%) is approximately 9 times the makespan of the first UL (12.5%).

Table 5.13: Makespan results for the disk-bound benchmark on AMD and Intel hosts

AMD		*Intel*	
UL	*makespan*	*UL*	*makespan*
12.5%	234.7±5.0	12.5%	36.0±0.9
25.0%	547.7±29.0	25.0%	74.0±1.5
37.5%	839.4±29.2	37.5%	114.7±2.4
50.0%	1397.8±184.5	50.0%	154.5±3.2
62.5%	1644.0±184.9	62.5%	195.9±3.8
75.0%	2006.3±182.7	75.0%	239.0±5.1
87.5%	2514.9±367.8	87.5%	279.1±25.8
100.0%	2571.5±147.2	100.0%	313.6±4.0

Figure 5.13 presents a graphic comparison of makespan for both hosts. The graph shows that the derivative is approximately constant in both hosts, indicating that the degradation vary proportionally with the UL, mainly on Intel host.

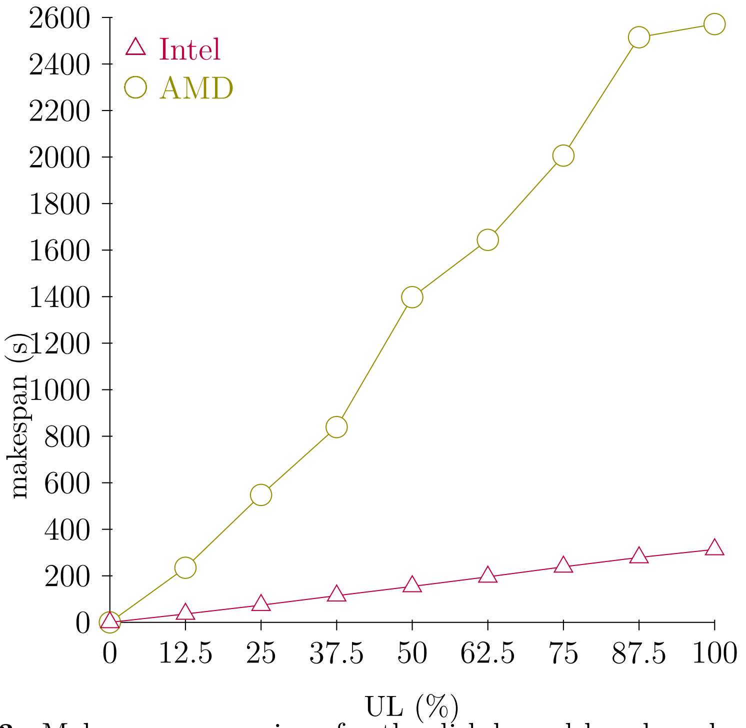


Figure 5.13: Makespan comparison for the disk-bound benchmark on AMD and Intel hosts

5.3 Energy efficiency analysis

This section analyzes the energy efficiency from the collected measurements. The energy efficiency metric defined in Equation 5.1 allows comparing the PC results for different ULs and hosts, while taking into account the execution time of an application. The lower the metric value, the higher energy efficiency of the host (lower power consumption and lower makespan).

$$eff = \frac{PC \times makespan}{number\ of\ instances \times 3600} \tag{5.1}$$

Table 5.14 reports the average energy efficiency for all ULs and both hosts. The minimum *eff* value for a given host and type of benchmark, which indicates the more efficient UL, is presented in bold font.

Table 5.14: Efficiency for different ULs on AMD and Intel hosts

UL	*AMD*			*Intel*		
	CPU	*mem.*	*disk*	*CPU*	*mem.*	*disk*
12.5%	0.798	1.800	**4.090**	0.403	0.403	**0.225**
25.0%	0.426	1.510	4.800	0.192	0.192	0.233
37.5%	0.299	1.430	4.910	0.159	0.227	0.240
50.0%	0.234	1.390	6.130	0.109	**0.185**	0.246
62.5%	0.196	**1.340**	5.780	0.089	0.266	0.256
75.0%	0.171	1.410	5.890	**0.081**	0.216	0.265
87.5%	**0.152**	1.570	6.340	0.083	0.236	0.265
100.0%	0.154	1.670	5.670	0.107	0.238	0.263

Results from the study indicate that the CPU-bound benchmark is more efficient at high ULs, the memory-bound benchmark is more efficient at medium ULs, and the disk-bound benchmark is more efficient at low ULs. These results hold for both hosts. Overall, Intel host is more efficient than AMD for all ULs and all types of benchmarks. This observation is coherent with reported comparison of these processors (CPUBOSS, 2014), where Intel host processor is presented as 10 times more efficient than AMD host processor. However, the energy analysis of this thesis is more comprehensive with respect the factors involved in the power consumption , because it considers more components of each host, not only the processor.

Finally, the high-critic UL (100%) is less efficient than the high-medium UL (87.5%) in all cases, except for disk-bound benchmark executions. For disk-bound benchmarks there are small variations on the PC values between ULs. Thus, the proposed efficiency metric improves when the number of instances increase.

5.4 Concluding remarks

The empirical study presented in this chapter was aimed at extracting the characteristics of the power consumption of multi-cores and their relationship with the type of workload, considering all the components involved holistically.

The single executions allowed concluding that the EC vary in different ways in dependency with the computing resource considered. In particular, the results showed that the EC is directly proportional to the UL in the CPU-bound workload. For the memory-bound workload, the single executions showed a deceleration in EC variation as the UL increases.

The combined execution of different types of benchmarks indicated that it is possible to gain EC with respect single executions, by the consolidation of task, taking advantage of the optimum UL (regarding EC) of computing resources. The results of two combined executions showed a remarkable gain in Intel host, where the difference in EC reached 57.65 Watts at UL (25%,25%). This gain represents the 32% of the EC of single executions. The three combined executions presented a EC gain on Intel host for all ULs regarding to single executions. However, on AMD host the EC is equal at (25 %, 25 %,25 %) and at (25 %, 25 %,50 %), and it is greater than EC of single executions at (50 %, 25 %,25 %) and at (25 %, 50 %,25 %).

On the one hand, the performance analysis showed that the CPU-bound workload did not degraded as the UL increases. On the other hand, results showed performance degradation of memory-bound workload and disk-bound workload, specially in AMD host.

The energy efficiency study, which combines results of power consumption and performance experiments, showed that the optimum UL depends on the type of workload: CPU-bound workload presented best efficiency at high UL, memory-bound workload at medium UL, and disk-bound workload at low UL. The difference between host efficiency observed in results was consistent with the published specifications of the hosts.

The numerical results of the presented power characterization can be used to build power consumption models. In addition, it is possible to consider the conclusions of the analysis as a guide to develop energy-aware scheduling strategies based on heuristics. The Chapter 6 presents the proposed power co-sumption models, several scheduling strategies, and the simulations performed for their evaluation.

Chapter 6

Models construction and simulation results

This chapter presents the utilization of power and performance characterization results, by building a wide variety of models, which are used in simulations to evaluate energy aware scheduling strategies. Section 6.1 presents several power consumption models built for independent and combined type of experiments. Then, Section 6.2 presents models built from data of performance evaluation. Finally, Section 6.3 presents the details and results of the performed simulations to evaluate scheduling strategies according to their energy benefits.

6.1 Power consumption models construction

This section presents the proposed power consumption models, obtained by applying polynomial regression over the power characterization results. Subsection 6.1.1 presents an insight about the proposed power consumption models. Subsection 6.1.2 introduces the power consumption models constructed from independently experiments. Then, Subsection 6.1.3 presents the power consumption models constructed from two combined resource experiments. Finally, Subsection 6.1.4 presents the power consumption models constructed from three combined resource experiments.

6.1.1 An insight of the proposed power consumption models

The models of power consumption proposed in this thesis are versions of specific application models. In these versions, the applications are partitioned into equivalence classes according their bounding computing resource (two applications are of the same class if they are intensive in the same computing resource). The identified variables that affect the energy consumption are: the computer resource involved, the load of the computer resource, and the total load of the host. All these variables are synthesized in the definition of UL. Equation 6.1 shows the general form of the power consumption models built, where x_n is the percentage of each resource running in the host and ϵ is the power consumption estimated by the model. The expression $x_1, x_2, ..., x_n$ corresponds to the definition of UL.

$$f(x_1, x_2, ..., x_n) \rightarrow \epsilon,\ x \in [0, 100] \wedge \epsilon \in \mathbb{R} \tag{6.1}$$

The models presented below seek the form of the function $f(x_1, x_2, ..., x_n)$ in each type of experiments performed in this thesis, which are: executions independent , combined executions of two computing resources, and combined executions of three computing resources. The models built from independent executions focused in the effective consumption (EC) because it contains more synthetic information than power overall power consumption (PC), since PC is just EC plus a constant. The models built from combined executions were build for PC.

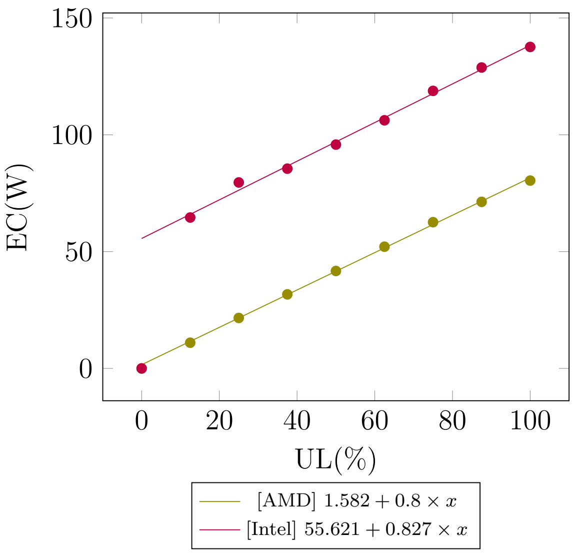
6.1.2 EC models considering a single resource

The models introduced in this subsection consider each computing resources individually. In this case, the proposed equation for the power consumption models (Equation 6.1) is instantiated as shown by the Equation 6.2, where the domain of function $f(x)$ has only one dimension.

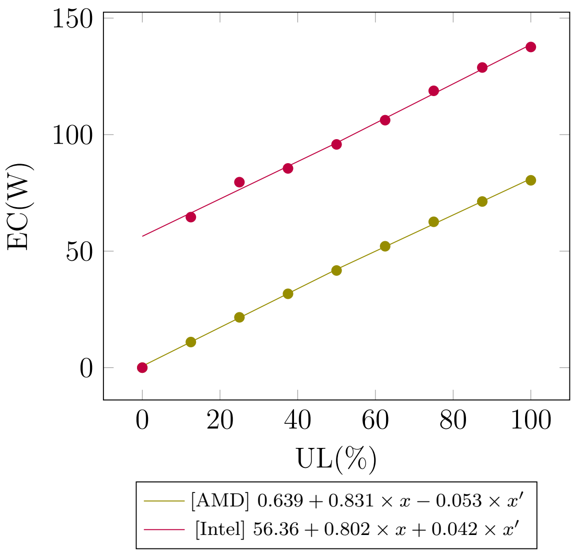
$$f(x) \rightarrow \epsilon,\ x \in [0, 100] \wedge \epsilon \in \mathbb{R} \tag{6.2}$$

To construct the energy models, the experimental data were adjusted by polynomial regression (Peckov, 2012). In order to compare different models, four formulas for function $f(x)$ were considered: linear, piece-wise, quadratic, and cubic. Two intervals were considered for the piece-wise model: [0,50] and [50,100]. The computing resources considered were CPU, memory and disk.

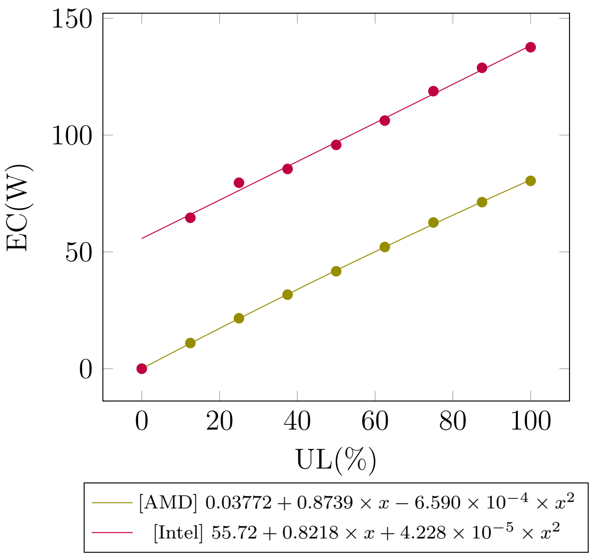
Graphs in Figure 6.1 present the four EC models for CPU-bound workload, on both AMD and Intel hosts. Graph legends show the equations for each curve, where x is the independent variable, i.e. the UL introduced in Section 4.2. The ancillary variable x', used in piece-wise model (Figure 6.1b), is zero when $x < 50$ and $x - 50$ in other case.



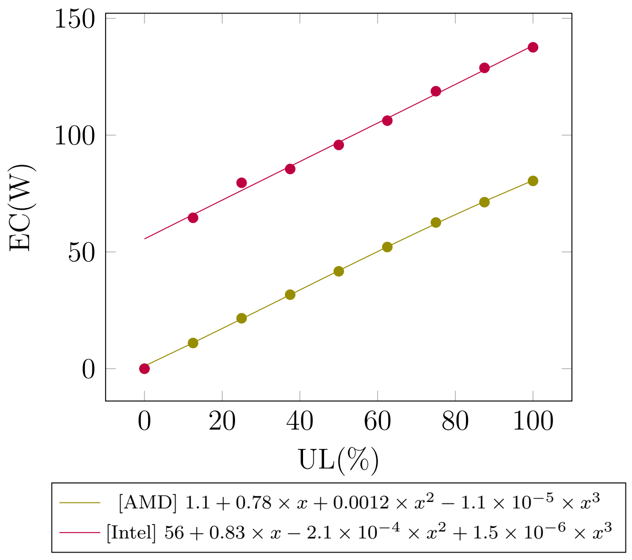
(a) Linear



(b) Piecewise



(c) Quadratic



(d) Cubic

Figure 6.1: EC models for CPU-bound workloads

A linear behavior of the models is observed. Besides, models with degree greater than one (i.e., quadratic and cubic) have no-linear coefficients close to zero, which indicates the linear dependency too. Both hosts have a similar rate of increase. For example, in linear models, the angular coefficient is 0.8 in AMD host and 0.827 in Intel host. This similarity indicates that the proportional dependency between UL and EC is independent of the architecture, for CPU-bound workloads.

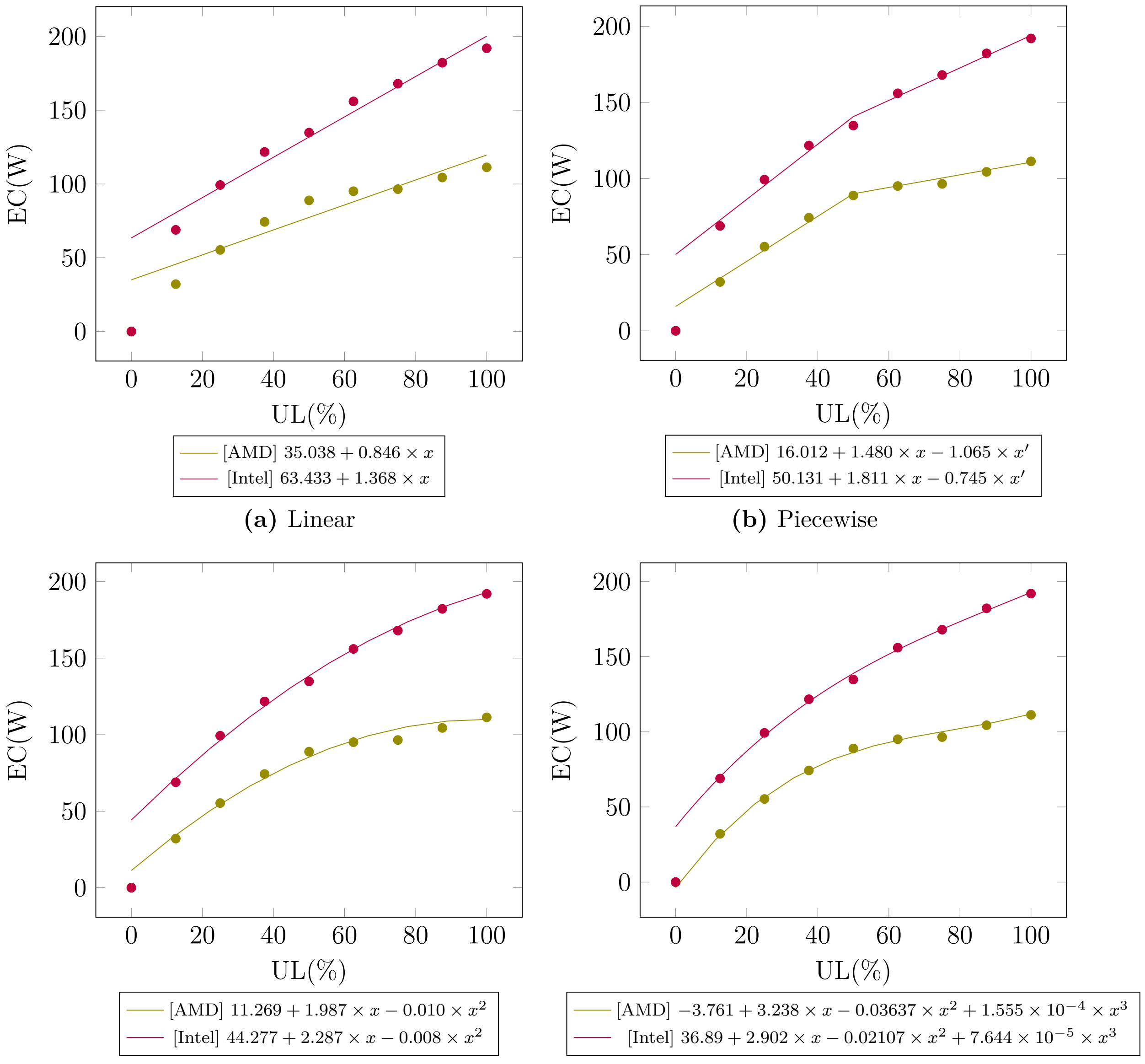
Table 6.1 reports the comparison of R^2 and $\bar{R}^2$ metrics for EC of CPU-bound workloads on both AMD and Intel hosts. All models present statistic values close to one, which indicates an acceptable fit to the data. However, quadratic and cubic models achieve a minor improvement over other approaches on AMD host. Piecewise model is the best for Intel host. AMD models show a better fit to the data than Intel models.

Table 6.1: Statistics of EC models for CPU-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.99937	0.99927	0.99584	0.99514
Piecewise	0.99964	0.99950	$\boxed{0.99599}$	$\boxed{0.99439}$
Quadratic	0.99979	9.99971	0.99584	0.99417
Cubic	$\boxed{0.99986}$	$\boxed{0.99976}$	0.99584	0.99272

Graphs in Figure 6.2 show models of the EC for the memory-bound workload on both AMD and Intel host. By comparing the angular coefficients of linear model it is possible to conclude that the EC on Intel increases faster than AMD, by a factor of 1.6 (1.368/0.846). The piece-wise graph for AMD (Figure 6.2b) shows a noticeable decrease of the first derivative in UL 50%, possibly due to the degradation of the performance caused by the waits in the access to memory, which implies less use of memory and, consequently, less consumption of energy.

Table 6.2 reports the comparison of R^2 and $\bar{R}^2$ metrics for EC models of memory-bound workload on both AMD and Intel hosts. The results indicate that cubic models achieves the best fit on AMD host as well as on Intel host. Piece-wise models present significant improvement regarding linear models on both AMD and Intel hosts.



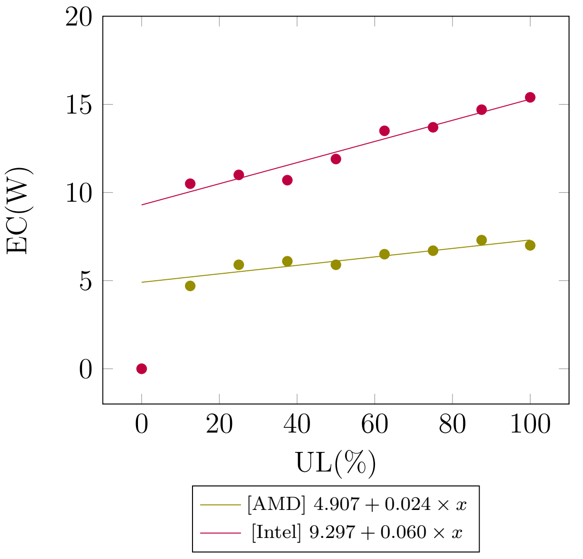
(a) Linear **(b)** Piecewise **(c)** Quadratic **(d)** Cubic

Figure 6.2: EC models for memory-bound workloads

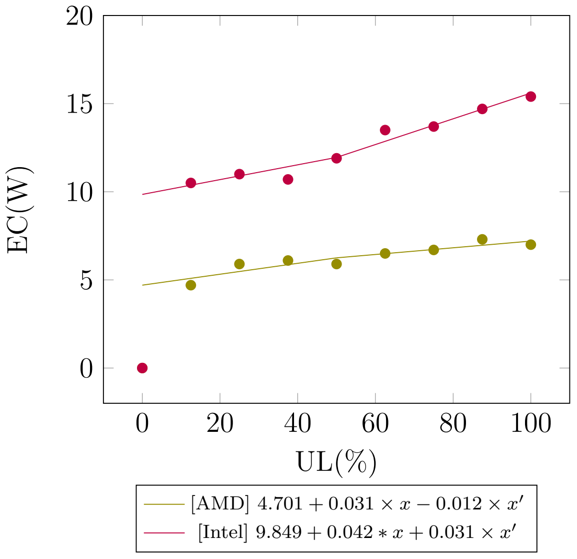
Table 6.2: Statistics of EC models for memory-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.90313	0.90310	0.97490	0.97072
Piecewise	0.99252	0.98952	0.99294	0.99011
Quadratic	0.98430	0.97802	0.99666	0.99533
Cubic	0.99483	0.99096	0.99772	0.99600

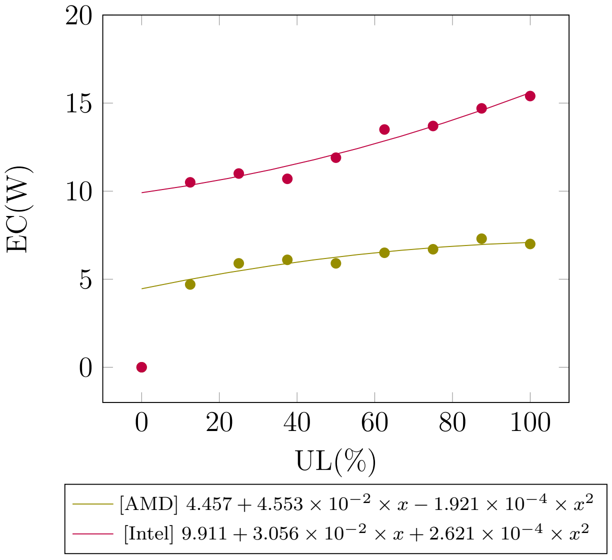
Graphs in Figure 6.3 present the EC models for the disk-bound workload, considered independently, on both AMD and Intel hosts. The angular coefficient of the linear models, which is close to zero, indicates that the EC is almost constant for disk-bound workload. Despite the constant behavior, a small increase is observed on Intel host, which does not occur on AMD host.



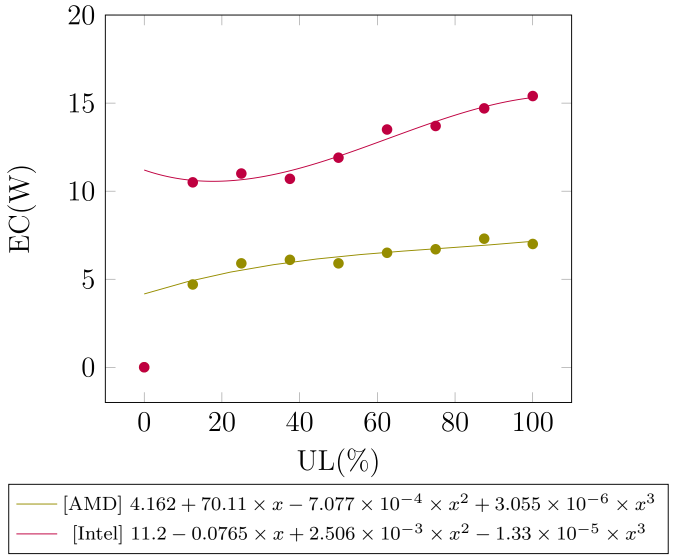
(a) Linear



(b) Piecewise



(c) Quadratic



(d) Cubic

Figure 6.3: EC models for disk-bound workloads

Table 6.3 reports the comparison of R^2 and $\bar{R}^2$ metrics of EC models of the disk-bound workload on both host. The results show that cubic models achieve the best fit regarding other approaches on both hosts. In this case, the models for Intel host present metrics values close to one, indicating a better fit to experimental data than on AMD host.

Table 6.3: Statistics of EC models for disk-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.84465	0.81875	0.94454	0.93530
Piecewise	0.85695	0.79973	0.96016	0.94423
Quadratic	0.87968	0.83016	0.95579	0.93811
Cubic	0.88344	0.79602	0.97179	0.95063

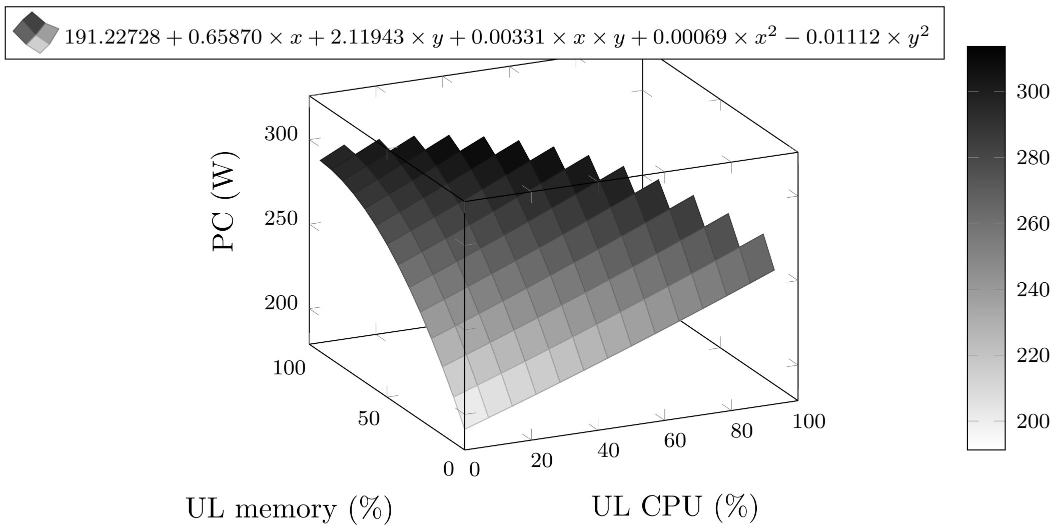
6.1.3 PC models considering two resources

In real systems, is usual to find several types of applications executing together in the same host. Moreover, the study presented in Chapter 5 indicates that it is possible to have an energy reduction in combined executions (possibly due to the use of the computing resources at optimal energy levels). This subsection considers models with two computing resource types as domain. For each pair of computing resources considered in the power characterization study of this thesis (i.e., CPU-memory, CPU-disk, and memory-disk), two two-dimensional models were constructed from the experimental data: a linear regression and a quadratic regression. The other approaches were not considered because the complexity introduced by the amount of terms of its equations, and also considering the high quality achieved by linear and quadratic approach. Equation 6.3 shows the linear PC models for combined CPU-memory experiments on AMD host and Equation 6.4 shows the same model on Intel host. Variable α is the UL of CPU-bound workload and β is the UL of memory-bound workload. Constructed models assume that the host has a minimum load, different to zero. The equations of the models indicate for both hosts that an increase in the memory-bound UL (i.e., variable β) has greater impact on PC than the same increase in the CPU-bound UL (i.e., variable α), since the coefficient of β is greater than the coefficient of α. Also, the model for Intel host increases faster than AMD models on both directions (CPU and memory).

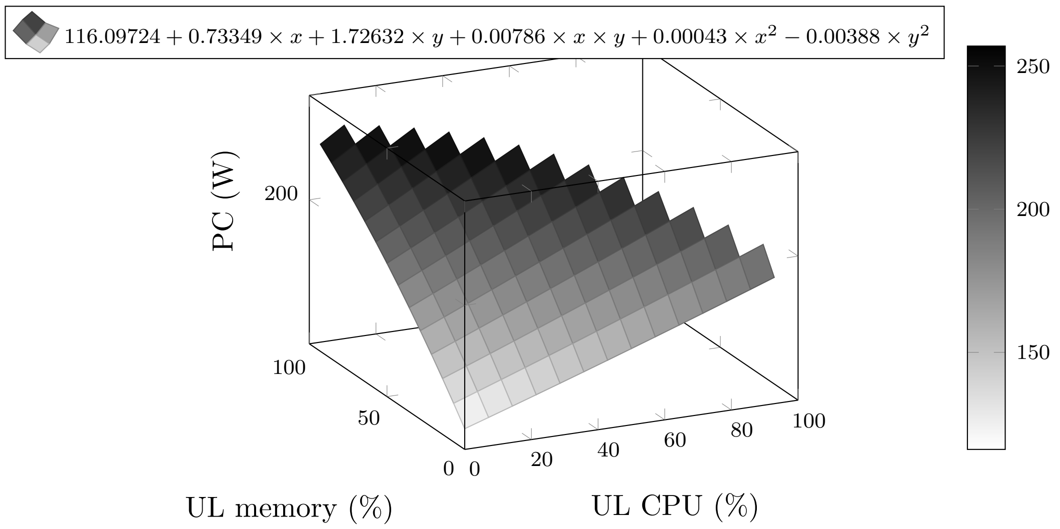
$$PC_{AMD} = 0.71923 \times \alpha + 1.17596 \times \beta + 201.69767 \quad (6.3)$$

$$PC_{Intel} = 0.89041 \times \alpha + 1.51675 \times \beta + 115.52924 \quad (6.4)$$

Figure 6.4a presents the quadratic regression model built from combined experiments of PC measurements on AMD host, for CPU and memory bound workloads executing together. Figure 6.4b shows the quadratic regression model on Intel host.



(a) AMD



(b) Intel

Figure 6.4: Quadratic PC models for combined CPU-memory workloads

Graphs legends show the equations for each curve, where variable x is the

CPU-bound workload UL and y is the memory-bound workload UL.

Table 6.4 presents the statistics of combined CPU and memory bound power consumption models. Results indicate that the quadratic model achieves a better fit to the data. The improvements of the quadratic approach over the linear approach is more notorious on AMD host than on Intel host.

Table 6.4: Statistics of PC models for combined CPU-memory workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.83832	0.80892	0.94252	0.93207
Quadratic	0.99156	0.98629	0.99423	0.99062

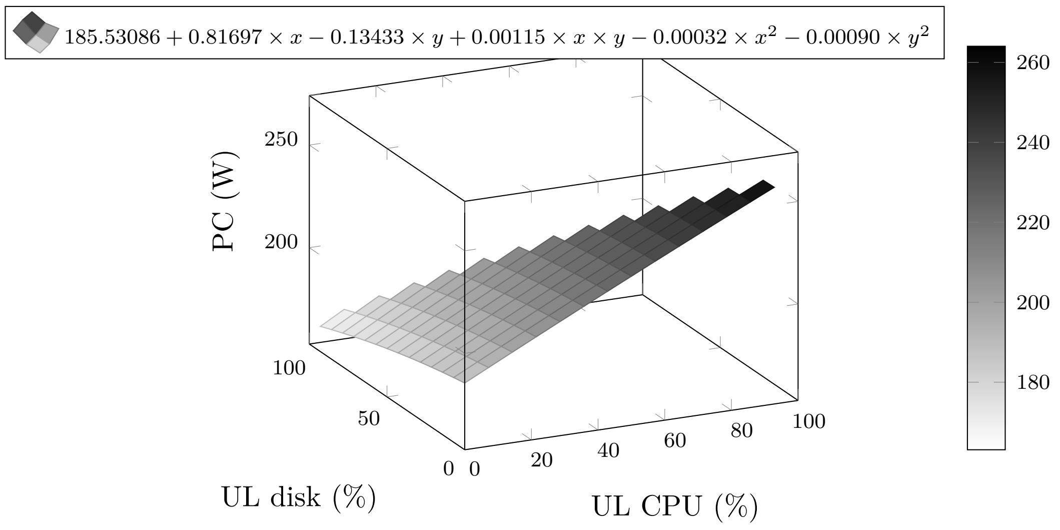
Equation 6.5 shows the linear PC models for combined CPU-disk experiments on AMD host and Equation 6.6 shows the same model on Intel host. Variable α is the UL of CPU-bound workload and variable γ is the UL of disk-bound workload. The coefficient of variable γ is almost zero, which indicates that the increase of disk-bound workload does not impact on PC.

$$PC_{AMD} = 0.80309 \times \alpha + 0.07151 \times \gamma + 186.30660 \tag{6.5}$$

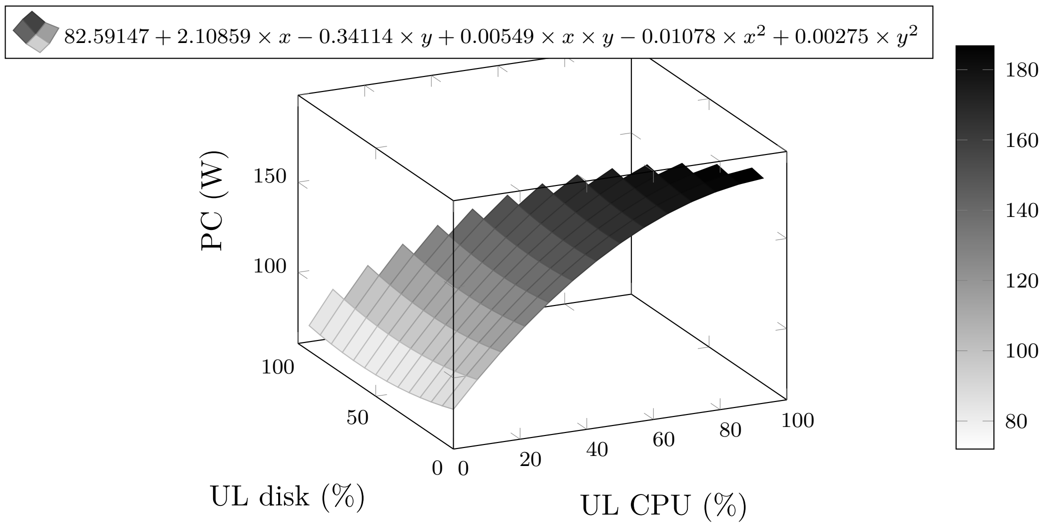
$$PC_{Intel} = 1.24615 \times \alpha - 0.05342 \times \gamma + 88.88702 \tag{6.6}$$

Figure 6.5a presents the quadratic regression model built from combined experiments of PC measurements on AMD host, for CPU and disk-intensive application executing together. Figure 6.5b shows the quadratic regression model on Intel host. Graphs legends show the equations for each curve, where variable x corresponds to the CPU workload UL and variable y corresponds to the disk workload UL. The almost zero absolute value of non-linear coefficients (i.e. coefficients of variables x^2 and y^2) of AMD model indicate a linear behaviour of PC.

Table 6.5 presents the statistics of combined CPU and disk-bound models. The results indicate that the quadratic model achieves a better fit to the data in both host. However, the linear model in AMD achieves a R-squared metric similar to that of the quadratic model, which confirms the observation about the linear behaviour of PC of combined CPU-disk bound workload on AMD.



(a) AMD



(b) Intel

Figure 6.5: Quadratic PC models for combined CPU-disk workloads

Table 6.5: Statistics of PC models for combined CPU-disk workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.99337	0.99216	0.8797	0.85783
Quadratic	0.99885	0.99814	0.96578	0.94439

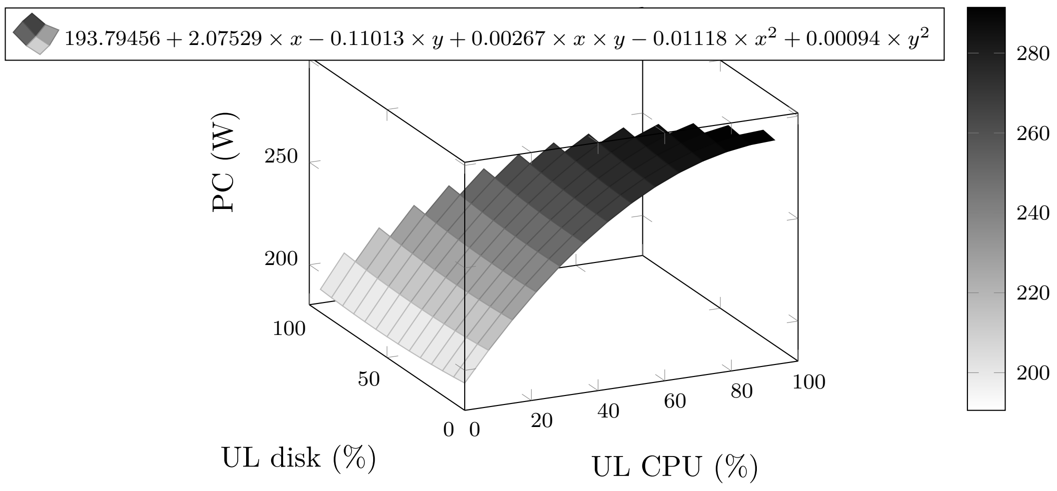
Equation 6.7 shows the linear models for combined memory-disk experiments on AMD host and Equation 6.8 shows the same model on Intel host.

The variable β is the UL of memory-bound workload and γ is the UL of disk-bound workload. The coefficient of β indicates that the Intel model increase faster than AMD model when memory UL increases and the coefficient of γ indicates that disk-bound workload has not impact on PC of both hosts.

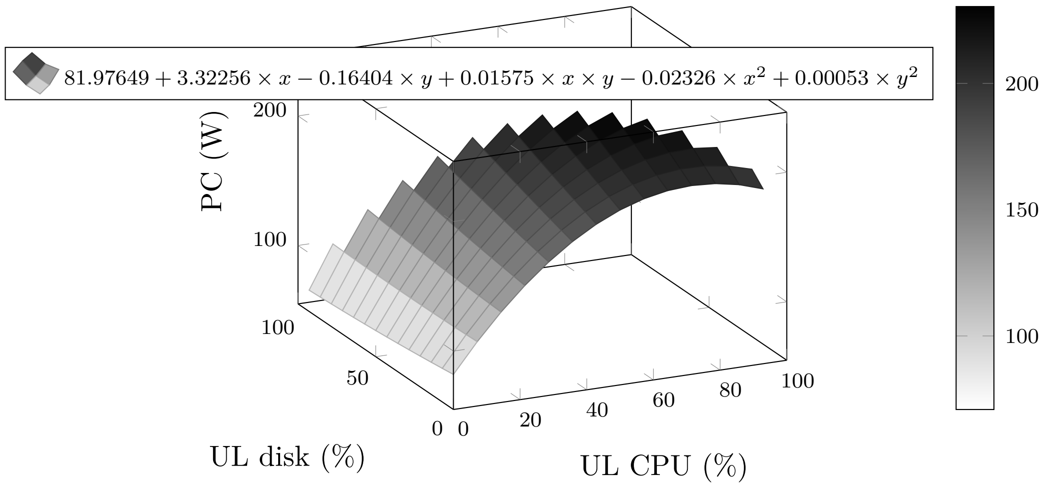
$$PC = 1.11686 \times \beta - 0.03832 \times \gamma + 204.40903 \tag{6.7}$$

$$PC = 1.50079 \times \beta + 0.03613 \times \gamma + 99.70075 \tag{6.8}$$

Figure 6.6a presents the quadratic regression model built from combined experiments of power consumption measurements on AMD host, for memory and disk intensive workloads executing together.



(a) AMD



(b) Intel

Figure 6.6: Quadratic PC models for combined memory-disk workloads

Figure 6.6b shows the quadratic regression model on Intel host, for memory and disk intensive workloads executing together. Graphs legends show curves equations, where variable x corresponds to the memory workload UL and variable y corresponds to the disk workload UL.

Table 6.6 presents the statistics of PC models of combined memory and disk intensive. The results indicate that the quadratic model achieves better fit to the data in both host.

Table 6.6: Statistics of PC models for combined memory-disk workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.89327	0.87387	0.66066	0.59896
Quadratic	$\boxed{0.99061}$	$\boxed{0.98475}$	$\boxed{0.96316}$	$\boxed{0.94014}$

6.1.4 PC models considering three resources

This subsection presents the most complete PC models proposed in this thesis, since they considers in a a combination of the three computational resources analyzed. The three-dimensional models built are classified in three types:

- Linear combination of the best independent models.
- Linear regression from three combined experiments of power consumption measurements.
- Cubic regression from three combined experiments of power consumption measurements.

The proposed models are described in the following paragraphs.

Linear combination of the best independent models. In the linear combination approach, the independent models presented in Subsection 6.1.2 are combined in order to build a three-dimensional model that considers the three type of workloads (CPU-bound, memory-bound, and disk-bound) as domain. Only the best independent model according statistics, was chosen between the four models built for each type of resource and host (which were reported in Tables 6.1, 6.2, 6.3).

Equation 6.9 presents the model of PC built applying the linear combination for CPU, memory and disk workload for AMD host, based on the best independent models. Since the independent models correspond to the EC modeling, the host IC is added in the PC models.

$$\begin{aligned}
&PC_{AMD} = 0.78 \times \alpha + 1.2 \times 10^{-3} \times \alpha^2 - 1.124 \times 10^{-5} \times \alpha^3 \\
&+3.238 \times \beta - 3.637 \times 10^{-2} \times \beta^2 + 1.555 \times 10^{-4} \times \beta^3 \\
&+7.011 \times 10^{-2} \times \gamma - 7.077 \times 10^{-4} \times \gamma^2 + 3.055 \times 10^{-6} \times \gamma^3 \\
&+((1.124 - 3.761 + 3.055 \times 10^{-6})/3) + 183.4
\end{aligned} \tag{6.9}$$

Equation 6.10 presents the model of PC built applying the linear combination for Intel host for CPU, memory, and disk workload. The ancillary variable α' is zero when $\alpha < 50$ and $\alpha - 50$ in other case.

$$\begin{aligned}
&PC_{INTEL} = 0.802 \times \alpha + 0.042 \times \alpha' \\
&+2.902 \times \beta - 2.107 \times 10^{-2} \times \beta^2 + 7.644 \times 10^{-5} \times \beta^3 \\
&-7.645 \times 10^{-2} \times \gamma + 2.506 \times 10^{-3} \times \gamma^2 - 1.330 \times 10^{-5} \times \gamma^3 \\
&+((56.362 + 36.89 + 11.20)/3) + 57
\end{aligned} \tag{6.10}$$

In both equations, α is the percentage of CPU-bound workload, β is the percentage of memory-bound workload, and γ is the percentage of disk-bound workload. The linear combination models assume that CPU-bound, memory-bound and disk-bound jobs arrive to the host (to be processed) with equal probability (0.33).

Linear regression from three combined experiments: In this approach, the models are constructed by fitting the data collected in three combined experiments of power measurements to a linear function, using the least squares method.

Equation 6.11 presents the linear model built from data of combined experiments of three on AMD host. The coefficients of the model indicate that the memory-bound workload is the one that most affects the PC.

$$PC_{AMD} = 186.82096 + 0.86923 \times \alpha + 1.97285 \times \beta + 0.17123 \times \gamma \tag{6.11}$$

Equation 6.12 presents the linear model built from data of three combined experiments on Intel host. On Intel host, the memory-bound workload is also the most important in terms of PC, since the model has its greatest growth in the direction of the variable β.

$$PC_{INTEL} = 102.65586 + 1.01680 \times \alpha + 2.12789 \times \beta - 0.16982 \times \gamma \tag{6.12}$$

Cubic regression from three combined experiments: In the case of the cubic regression application, the models are constructed by fitting the data collected in three combined experiments of power measurements to a cubic function, using the least squares method.

Equation 6.13 presents the cubic model built from data of three combined experiments on AMD host.

$$\begin{aligned}
&PC_{AMD} = 0.00139 + 0.01735 \times \alpha + 0.01738 \times \beta + 0.01734 \times \gamma \\
&+0.14433 \times \alpha \times \beta + 0.14420 \times \alpha \times \gamma + 0.14429 \times \beta \times \gamma \\
&+0.28924 \times \alpha^2 + 0.28970 \times \beta^2 + 0.28905 \times \gamma^2 - 0.00643 \times \alpha^2\beta \\
&-0.00971 \times \alpha^2 \times \gamma - 0.01113 \times \beta^2 \times \alpha - 0.01206 \times \beta^2 \times \gamma \\
&-0.00784 \times \gamma^2 \times \beta - 0.00549 \times \gamma^2 \times \beta + 0.01189 \times \alpha \times \beta \times \gamma \\
&+0.00152 \times \alpha^3 + 0.00366 \times \beta^3 + 0.00052 \times \gamma^3
\end{aligned} \tag{6.13}$$

Equation 6.14 presents the cubic model built from data of three combined experiments on Intel host.

$$\begin{aligned}
&PC_{INTEL} = 0.00090 + 0.01129 \times \alpha + 0.01130 \times \beta + 0.01123 \times \gamma \\
&+0.09400 \times \alpha \times \beta + 0.09367 \times \alpha \times \gamma + 0.09375 \times \beta \times \gamma \\
&+0.18814 \times \alpha^2 + 0.18844 \times \beta^2 + 0.18710 \times \gamma^2 - 0.00123 \times \alpha^2 \times \beta \\
&-0.00943 \times \alpha^2 \times \gamma - 0.00437 \times \beta^2 \times \alpha - 0.01043 \times \beta^2 \times \gamma \\
&+0.00132 \times \gamma^2 \times \beta + 0.00346 \times \gamma^2 \times \beta - 0.00232 \times \alpha \times \beta \times \gamma \\
&+0.00120 \times \alpha^3 + 0.00245 \times \beta^3 - 0.00315 \times \gamma^3
\end{aligned} \tag{6.14}$$

Table 6.7 reports the comparison of R^2 and $\bar{R}^2$ metrics of PC models (linear and cubic) for combined CPU-memory-disk on both hosts.

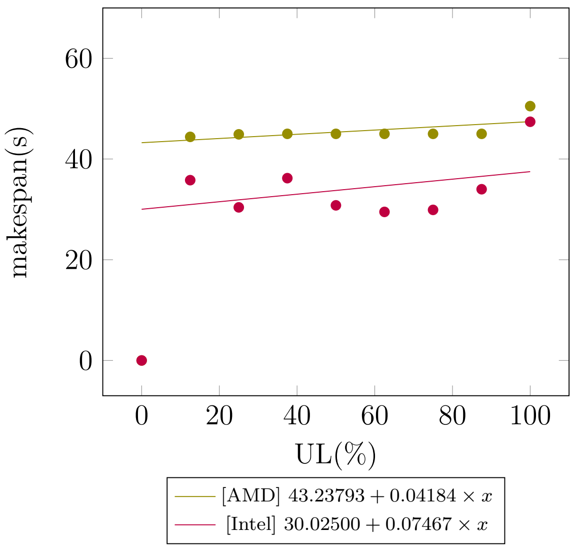
Table 6.7: Statistics of PC models for combined CPU-memory-disk workloads

	AMD		*Intel*	
Linear	0.98793266	0.981899	0.75077941	0.62616912
Cubic	**1**	**1**	**1**	**1**

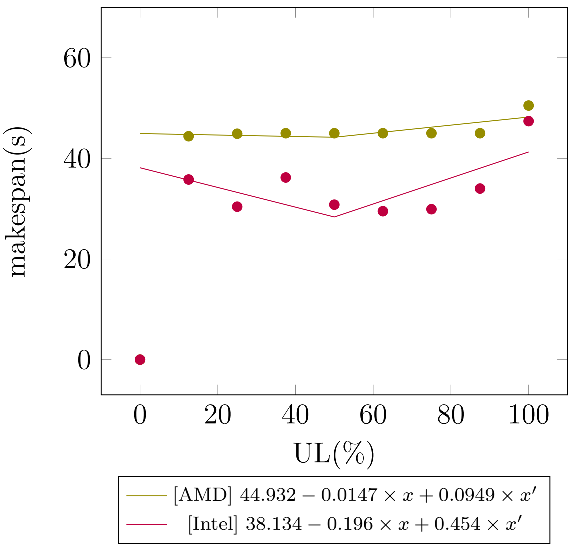
6.2 Performance models construction

This section introduces the performance models built from experimental results, presented in Section 5.2. In this case, the makespan reported by the performance experiments is adjusted using the least squares to linear, piece-wise, quadratic, and cubic functions. As in the models of power consumption, the interval considered for the piece-wise model were [0,50] and [50,100].

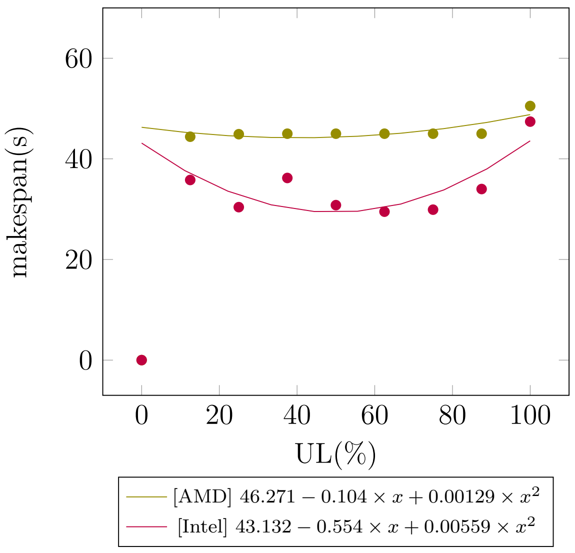
Figure 6.7 presents the performance models for CPU-bound workloads on both AMD and Intel hosts.



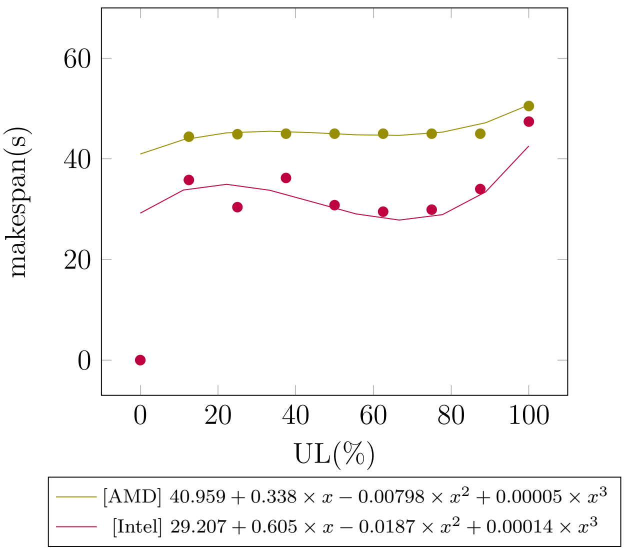
(a) Linear



(b) Piecewise



(c) Quadratic



(d) Cubic

Figure 6.7: Performance models for CPU-bound workloads

Graphs legends show equations of each curve, where x is the independent variable, i.e. the utilization level UL introduced in Section 4.2.4. The ancillary variable x', used in piece-wise model (Figure 6.1b), is zero when $x < 50$ and $x - 50$ in other case. The angular coefficient of linear model is almost zero, which indicates that the makespan of the CPU-bound workload is not affected by the UL of the host.

In addition, the fact that the same benchmark is used on both hosts indicates that Intel host is approximately 13% faster than AMD host, considering the CPU-bound workload.

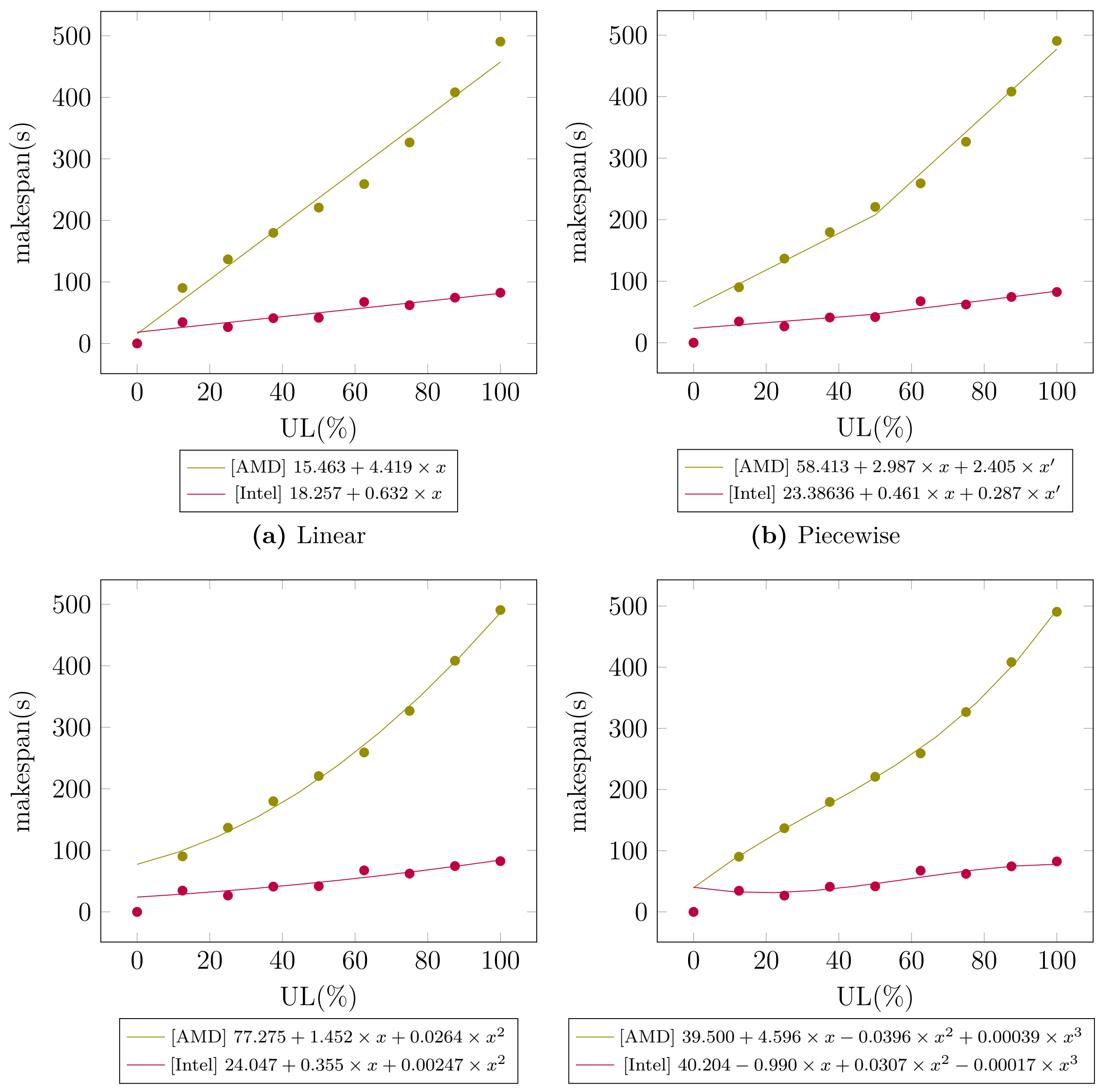
Table 6.8 reports the comparison of R^2 and $\bar{R}^2$ metrics for performance models of the CPU-bound workload on both AMD and Intel hosts. In this case, cubic models achieve the best fit on both hosts, having notorious improvements regarding to the other approaches.

Table 6.8: Statistics of performance models for CPU-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.40798	0.30931	0.14784	0.0058097
Piecewise	0.53887	0.35441	0.48883	0.28436
Quadratic	0.65197	0.51276	0.66616	0.53263
Cubic	0.89498	0.81622	0.85611	0.74819

Figure 6.8 presents the performance models for the memory-bound workload on both AMD and Intel hosts. The comparison of the angular coefficient of linear models indicates that indicate that makespan on the AMD host increases approximately seven times faster than on Intel host.

Table 6.9 reports the comparison of R^2 and $\bar{R}^2$ metrics for performance models of the memory-bound workload on both AMD and Intel hosts. Results indicate that the cubic model achieves the best fit on both hosts. On AMD host, the models are better fitted to the data than on Intel host, particularly for the piece-wise, quadratic, and cubic approach. In addition, the piece-model presents a better fit to the data than the quadratic model on Intel host.



(a) Linear

(b) Piecewise

(c) Quadratic

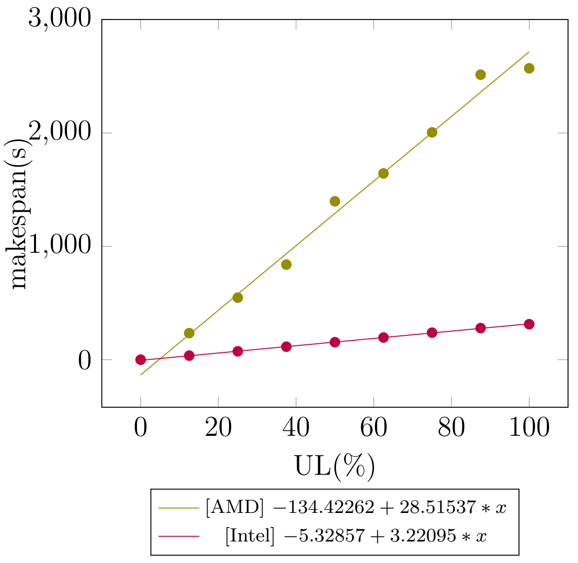
(d) Cubic

Figure 6.8: Performance models for memory-bound workloads

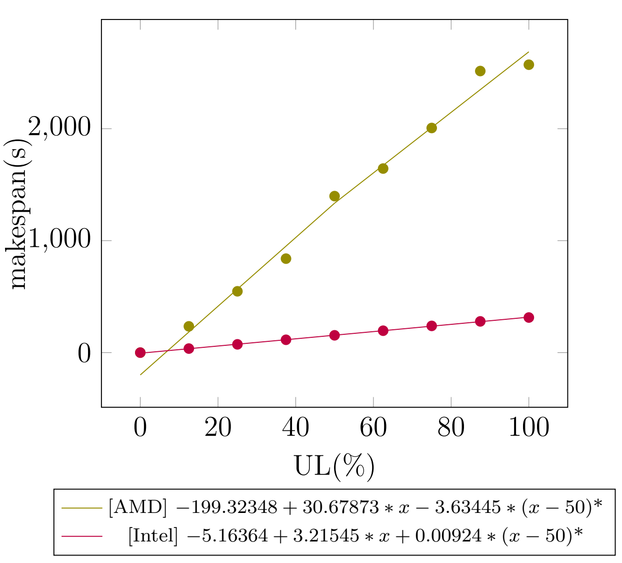
Table 6.9: Statistics of performance models for memory-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.97447	0.97022	0.89682	0.87963
Piecewise	0.99248	0.98947	0.90836	0.8717
Quadratic	0.99617	0.99464	0.90538	0.86753
Cubic	0.9988	0.9979	0.927	0.87224

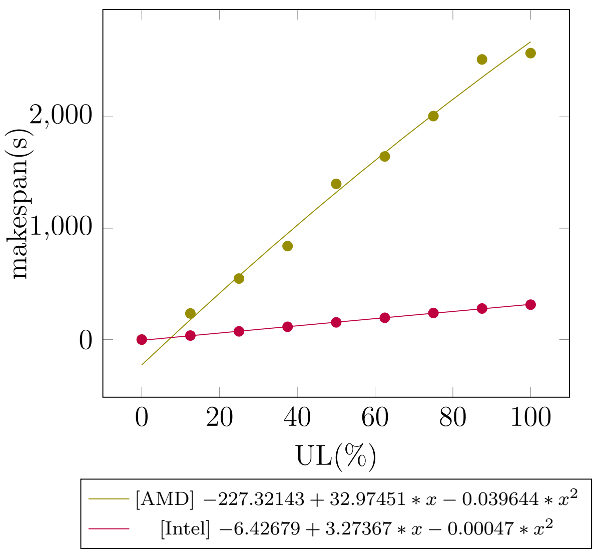
Figure 6.9 presents the performance models for the disk-bound workload on both AMD and Intel hosts. The angular coefficient of the linear models indicates that the increase of the makespan on AMD is nine times faster than the increase of the makespan on Intel host, that is, the degradation of the performance as the level of utilization increases is nine times greater.



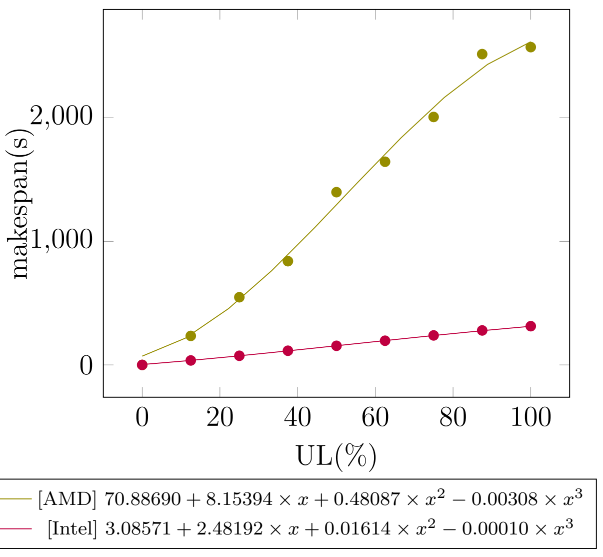
(a) Linear



(b) Piecewise



(c) Quadratic



(d) Cubic

Figure 6.9: Performance models for disk-bound workloads

Table 6.10 shows R^2 and $\bar{R}^2$ comparison for performance models of disk-bound workload on both AMD and Intel hosts. Results indicate that cubic models achieve the best fit on both hosts.

Table 6.10: Statistics of performance models for disk-bound workloads

	AMD		*Intel*	
Model	R^2	$\bar{R}^2$	R^2	$\bar{R}^2$
Linear	0.98768	0.98768	0.99958	0.99958
Piecewise	0.98868	0.98868	0.99958	0.99958
Quadratic	0.98887	0.98887	0.9996	0.9996
Cubic	0.99286	0.99286	0.99992	0.99992

6.3 Schedulers evaluation

This section describes the simulations performed for comparing different scheduling strategies regarding energy efficiency. Subsection 6.3.1 explains how the proposed models are implemented in a simulation tool. Then, Subsection 6.3.2 presents details of the simulation. Subsection 6.3.3 introduces the workload for simulation, based on real traces of HPC infrastructures. Subsection 6.3.4 presents the strategies used in scheduling. Finally, Subsection 6.3.5 presents the results of simulation, comparing the energy consumption reported by each considered strategy.

.

6.3.1 Energy model implementation

The proposed energy models were implemented in the version of Cloudsim developed by CICESE. Cloudsim handles the following main entities: hosts, which correspond to the physical host, virtual machines (VM), which are the processing units and assigned to the hosts, and jobs, which correspond to the workload. The jobs are assigned to the VMs to be processed.

Two Java classes were modified in Cloudsim: i) `PowerModelJobType`, located in package `cicese.cloudbus.cloudsim.power.models`, which is used by Cloudsim for scheduling tasks; and ii) `PowerModelJobType` located in package `cicese.cloudbus.cloudsim.util.power`, which is applied for computing the total power consumption in a post-processing stage, using the simulation results.

In each Java class, the method `getEnergy(double` α`,double` β`)` was extended to include the main features of each energy model developed in this thesis. This method returns the current power consumption of a VM that executes a task. The method `getEnergy(double` α`,double` β`)` was overrode to include the empirical coefficients from the linear interpolation and the corresponding value of IC for each host.

6.3.2 Simulation details

Two different power consumption models were considered, modeling multicore hosts with AMD and Intel architectures (eight cores each), which correspond to versions of the linear combination models presented on Equations 6.9 and 6.10. Since the simulation only considered two type of workload (CPU-bound and memory-bound), the term of variable γ (disk-bound workload) was eliminated from the equations.

Thirty different workloads were considered in the evaluation for each architecture, thus a total number of 60 simulations were performed (without considering the independent executions of the stochastic algorithms). In order to focus the simulations in the energy consumption study, some simplifications were considered: i) the physical hosts within a same simulation are identical; ii) only one VM is created per physical host, which can execute as many tasks as cores available in the host; and iii) each job has only one task, so the terms job and task both denote the atomic workload unit to schedule. These simplifications are realistic in the context of scientific computing infrastructures, for example Cluster FING (Nesmachnow, 2010).

During a simulation, N independent jobs arrive in different times (considering realistic distributions, see Subsection 6.3.3 for details) and the scheduler must deliver the job to one VM from M active VMs. Each VM can execute up to eight jobs at the same time and a job requires only one core to execute (1/8 of the VM capacity). When all active VMs are full, a new VM is turned on and this VM is considered active from that moment. It is assumed that a VM always has enough memory, bandwidth, and disk to execute any job.

6.3.3 Workloads description

The workloads used for the experimental evaluation are based on traces of real applications taken from HPC Parallel Workloads Archive executed on parallel supercomputers, clusters, and grids. The workloads include traces from DAS2-University of Amsterdam, DAS2–Delft University of Technology, DAS2–Utrecht University, DAS2–Leiden University, Royal Institute of Technology (KTH), DAS2–Vrije University Amsterdam, High Performance Computing Center North (HPC2N), Cornell Theory Cente (CTC), and (Los Alamos National Laboratory) LANL. A detailed study of the workloads were presented by Feitelson et al. (2014). The information included in these workloads allows evaluating allocation strategies over real scenarios and applications.

Workloads are specified in a format that extends the Standard Workload Format (swf). The extended format, introduced by Armenta-Cano et al. (2017), adds two new fields to the format: *codec utilization* and *job type*.

The fields of the extended swf format used in this thesis include:

- *Job id:* is the identifier of the job.
- *Submitted time:* is the arrival time of the job, in seconds. A job cannot begin execution before its submitted time.
- *Job length:* is a measure of the needed resources to complete the job, expressed in MIPS. When instantiated over a specific host, the duration of the job is given by the quotient of the job length and the power processing of the job (i.e., a job with length of 800 MIPS executing in a core of 100 MIPS will execute for eight seconds).
- *Codec utilization:* this field represents the percentage of a VM defined over the host that is requested to be used by a job. In the context of this research, codec represents the number of cores required by a job.
- *Job type:* this field allows specifying the type of the job (CPU-bound, memory-bound, disk-bound, or other relevant type). In the experiments performed in this article, type 0 represents a CPU-bound job and type 1 represents a memory-bound job.

6.3.4 Scheduling heuristics

The scheduling strategies were compared in experiments performed considering the two power consumption models over the 30 different workloads studied (W01, W02, . . ., W30). Each workload accounts for one week of operation of a real HPC platform, as described in subsection 6.3.3. The computing platform simulated in the experiments is composed of 150 multicore hosts with eight cores each.

An on-line scheduling approach is applied. On-line scheduling is a model in which the scheduling decisions are taken immediately after a job arrives or is released (Tchernykh et al., 2014). Although they are based on local and often sub-optimal decisions, on-line schedulers provide some advantages over static scheduling methods including a more realistic model of the user-system interaction and less information is needed to perform the resource assignment. In addition, on-line schedulers demand less processing for the scheduling procedure, which means less overhead. In the experiments reported in this section, the destination VM is chosen by the cloud scheduler by applying a specific strategy when a new job arrives to the system. Figure 6.10 illustrates the scheduling approach used in simulations.

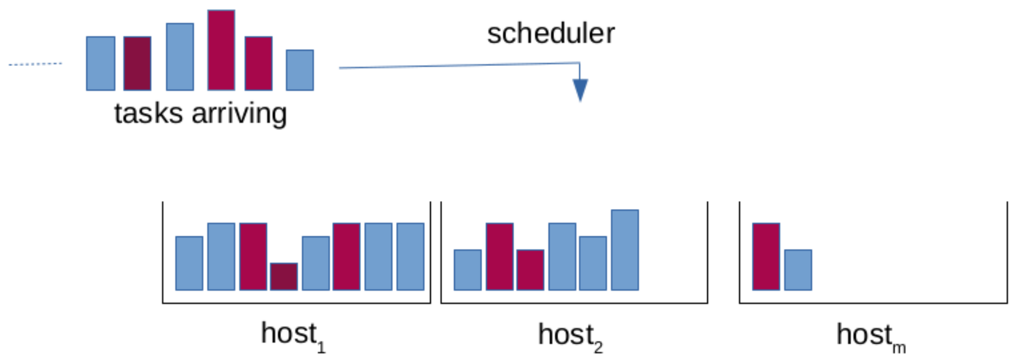


Figure 6.10: Scheduling approach

The proposed scheduling strategies hold a list of active VMs (the active list), i.e., VMs that are currently executing jobs. Adding a VM to the active list corresponds to turning on a physical host. A VM is capable of executing a job if the number of free processors of the VM is greater than (or equal) the number of processors that this job requires. Strategies are implemented using well-know heuristics for scheduling problems. The proposed heuristics are oriented to fulfill the followings goals: minimize the host utilization, maximize the host utilization, balance the utilization, and minimize the power consumption of each host.

The studied heuristics include:

- *Random* (RD). The purpose of the heuristic is to assign the jobs randomly. The destination VM for an arriving job is selected randomly from the list of active VMs, applying a uniform distribution.

- *Round Robin* (RR). The purpose of the heuristic is is to distribute equitably the jobs between the VMs, in a rational order (Leung et al., 2004; Silberschatz et al., 2012). The destination VM for an arriving job is selected in circular order. VMs are considered ordered by VM identifier and when the last active VM is assigned, the VM with identifier equal to 0 is assigned again. Algorithm 4 explains the Round Robin strategy, where *vm_idx* is the index of the VM where the last job was assigned.

Algorithm 4 Round Robin Strategy

```
 1: $first\_vm\_idx \leftarrow vm\_idx$
 2: $assigned \leftarrow false$
 3: repeat
 4:     $vm \leftarrow active\_vm\_list[vm\_idx]$;
 5:     if $is\_capable(vm, arrived\_job)$ then
 6:         $assign(vm, arrived\_job)$
 7:         $assigned \leftarrow true$
 8:     end if
 9:     $vm\_idx = vm\_index + 1 \pmod{size(active\_vm\_list)}$
10: until $assigned$ or $first\_vm\_idx = vm\_idx$
```

- *First Fit* (FF). The purpose of the heuristic is to assign jobs to the first possible VM (El-Rewini et al., 1994). The destination VM for an arriving job is the first VM in the active list with enough free resources (in this thesis, cores) as requested by the arriving job. The active list is considered to be ordered by VM identifier, ascendant.
- *Minimum Energy* (mE). The purpose of the heuristic is to assign jobs to the possible VM that consumes less power, according to the current assignment. The destination VM for an arriving job is the VM whose corresponding host has the lowest energy consumption. Algorithm 5 explains the details of the mE strategy.
- *Minimum Utilization* (mU). The purpose of the heuristic is to assign jobs to the possible VM that have the most resources available (in this thesis, available cores). The destination VM for an arriving job is the VM of the active list with the lowest percentage of utilization.
- *Maximum Utilization* (MU). The purpose of the heuristic is to assign jobs to the possible VM that have the less resources available (in this

thesis, available cores). The destination VM for an arriving job is the VM of the active list with the highest percentage of utilization.

Algorithm 5 Min. Energy Strategy

```
1: for all $vm$ in $active\_vm\_list$ do
2:   if $is\_capable(vm, arrived\_job)$ then
3:     if $is\_not\_asiigned(arrived\_job)$ then
4:       $assign(vm, arrived\_job)$
5:     else
6:       if $current\_power\_of\_host(vm) <$
         $current\_power\_of\_host(assigned\_vm\_(job))$ then
7:         $assign(vm, arrived\_job)$
8:       end if
9:     end if
10:  end if
11: end for
```

6.3.5 Simulation results

Table 6.11 reports the power consumption values for each scheduling strategy using the AMD energy model and Table 6.12 reports the power consumption values for each scheduling strategy using the Intel energy model. All values on the tables are reported in Watt per second ($\times 10^8$). All scheduling strategies are deterministic, except for RD, which is non-deterministic. For RD, the mean and standard deviation of power consumption values (calculated in 20 independent simulations performed for each workload) are reported.

Results in Tables 6.11 and 6.12 indicate that strategies that maximize the utilization of hosts (i.e., MU and FF) achieved better energy efficiency. Conversely, lower efficiency is achieved when the utilization is minimized (mU). Strategy mE, which is oriented to minimize the energy consumption also achieved worse results. The fact that both utilization model and energy model are linear explains the results for strategy Me, because minimizing energy corresponds to minimizing the utilization. Strategies that balance the utilization (RR, RD) achieved intermediate results regarding the other reported strategies.

Table 6.11: Total energy consumption of the studied scheduling strategies using the AMD energy model

Workload	RD	FF	RR	MU	mU	mE
W1	1.47±0.03	1.29	1.51	1.28	1.73	1.66
W2	1.47±0.02	1.26	1.41	1.27	1.60	1.58
W3	1.92±0.02	1.73	1.88	1.72	2.04	2.06
W4	2.32±0.03	2.12	2.31	2.11	2.50	2.48
W5	2.08±0.03	1.91	2.06	1.87	2.37	2.34
W6	1.77±0.03	1.61	1.79	1.61	1.98	1.90
W7	1.70±0.04	1.53	1.86	1.55	1.98	1.95
W8	2.14±0.04	1.95	2.09	1.95	2.34	2.25
W9	2.19±0.02	2.02	2.17	2.01	2.29	2.29
W10	1.73±0.03	1.58	1.79	1.54	1.91	1.83
W11	2.22±0.03	2.00	2.21	1.99	2.52	2.43
W12	2.08±0.04	1.87	2.14	1.86	2.31	2.28
W13	2.25±0.04	2.05	2.37	2.02	2.65	2.59
W14	1.91±0.02	1.81	1.91	1.80	2.06	2.08
W15	2.23±0.03	2.05	2.23	2.05	2.38	2.36
W16	1.62±0.03	1.47	1.51	1.45	1.81	1.76
W17	1.89±0.03	1.69	1.85	1.67	2.13	2.08
W18	1.59±0.03	1.48	1.57	1.46	1.69	1.65
W19	2.01±0.03	1.85	2.06	1.87	2.24	2.20
W20	2.37±0.05	2.08	2.40	2.07	2.58	2.41
W21	2.35±0.05	1.99	2.37	1.99	2.76	2.66
W22	2.32±0.05	1.86	2.32	1.89	2.65	2.49
W23	1.51±0.02	1.39	1.50	1.40	1.66	1.62
W24	1.83±0.02	1.70	1.90	1.70	1.95	1.97
W25	1.92±0.03	1.74	1.96	1.69	2.06	2.04
W26	2.29±0.04	2.05	2.25	2.02	2.48	2.39
W27	1.97±0.03	1.82	2.08	1.81	2.29	2.19
W28	2.13±0.04	1.96	2.12	1.89	2.30	2.32
W29	2.72±0.05	2.47	2.73	2.47	3.08	3.06
W30	2.48±0.03	2.18	2.47	2.21	2.74	2.79

Table 6.12: Total energy consumption of the studied scheduling strategies using the Intel energy model

Workload	RD	FF	RR	MU	mU	mE
W1	1.07±0.02	0.96	1.09	0.96	1.22	1.21
W2	1.07±0.01	0.94	1.04	0.95	1.15	1.11
W3	1.42±0.02	1.31	1.39	1.30	1.49	1.50
W4	1.66±0.01	1.55	1.66	1.54	1.77	1.76
W5	1.51±0.02	1.42	1.48	1.39	1.68	1.63
W6	1.26±0.02	1.16	1.27	1.16	1.38	1.35
W7	1.26±0.03	1.15	1.35	1.16	1.42	1.39
W8	1.57±0.01	1.46	1.54	1.46	1.69	1.62
W9	1.63±0.02	1.53	1.61	1.53	1.69	1.70
W10	1.27±0.02	1.18	1.30	1.16	1.37	1.34
W11	1.64±0.02	1.51	1.63	1.50	1.81	1.72
W12	1.54±0.02	1.42	1.58	1.41	1.68	1.66
W13	1.67±0.03	1.55	1.74	1.53	1.90	1.85
W14	1.45±0.02	1.39	1.45	1.38	1.54	1.55
W15	1.71±0.01	1.61	1.71	1.60	1.80	1.79
W16	1.19±0.02	1.11	1.10	1.10	1.30	1.29
W17	1.37±0.02	1.26	1.35	1.25	1.51	1.46
W18	1.18±0.01	1.11	1.17	1.11	1.24	1.24
W19	1.50±0.02	1.40	1.53	1.42	1.64	1.62
W20	1.71±0.03	1.54	1.73	1.54	1.83	1.85
W21	1.72±0.02	1.50	1.73	1.50	1.95	1.91
W22	1.66±0.02	1.40	1.66	1.41	1.86	1.77
W23	1.13±0.02	1.06	1.12	1.06	1.21	1.17
W24	1.37±0.01	1.30	1.42	1.29	1.45	1.45
W25	1.45±0.01	1.34	1.47	1.31	1.53	1.50
W26	1.71±0.02	1.57	1.69	1.55	1.83	1.75
W27	1.47±0.02	1.38	1.54	1.38	1.66	1.59
W28	1.56±0.02	1.46	1.55	1.42	1.66	1.67
W29	2.03±0.03	1.88	2.04	1.88	2.24	2.28
W30	1.86±0.02	1.68	1.86	1.70	2.02	2.04

A relevant analysis regarding energy efficient is the comparison of the best schedulers against strategies that apply a Business-as-Usual approach, i.e., when energy efficiency is not considered to perform the job-to-resources allocation. RR is one of the most widely used strategies for scheduling and it is included as the default scheduler in popular resource managers such as Maui and Condor Jackson et al. (2001). Maui and Condor is widely used by multiple computational platform such as custer, data centers, distributed platforms, among others.

The percent improvements (GAP) over RR of each proposed scheduling strategies are reported in Table 6.13. The percent improvements (GAP) over RR of each scheduling strategies are reported in Table 6.14.

Table 6.13: GAPs over RR of each scheduling strategies on AMD host

workload	FF	RD	MU	mU	mE
W1	14.57%	2.65%	15.23%	-14.57%	-9.93%
W2	10.64%	-4.26%	9.93%	-13.48%	-12.06%
W3	7.98%	-2.13%	8.51%	-8.51%	-9.57%
W4	8.23%	-0.43%	8.66%	-8.23%	-7.36%
W5	7.28%	-0.97%	9.22%	-15.05%	-13.59%
W6	10.06%	1.12%	10.06%	-10.61%	-6.15%
W7	17.74%	8.60%	16.67%	-6.45%	-4.84%
W8	6.70%	-2.39%	6.70%	-11.96%	-7.66%
W9	19.31%	-8.42%	-7.43%	0.50%	-13.37%
W10	11.73%	3.35%	13.97%	-6.70%	-2.23%
W11	9.50%	-0.45%	9.95%	-14.03%	-9.95%
W12	12.62%	2.80%	13.08%	-7.94%	-6.54%
W13	13.50%	5.06%	14.77%	-11.81%	-9.28%
W14	5.24%	0.00%	5.76%	-7.85%	-8.90%
W15	8.07%	0.00%	8.07%	-6.73%	-5.83%
W16	2.65%	-7.28%	3.97%	-19.87%	-16.56%
W17	8.65%	-2.16%	9.73%	-15.14%	-12.43%
W18	5.73%	-1.27%	7.01%	-7.64%	-5.10%
W19	10.19%	2.43%	9.22%	-8.74%	-6.80%
W20	13.33%	1.25%	13.75%	-7.50%	-0.42%
W21	16.03%	0.84%	16.03%	-16.46%	-12.24%
W22	19.83%	0.00%	18.53%	-14.22%	-7.33%
W23	7.33%	-0.67%	6.67%	-10.67%	-8.00%
W24	10.53%	3.68%	10.53%	-2.63%	-3.68%
W25	11.22%	2.04%	13.78%	-5.10%	-4.08%
W26	8.89%	-1.78%	10.22%	-10.22%	-6.22%
W27	12.50%	5.29%	12.98%	-10.10%	-5.29%
W28	7.55%	-0.47%	10.85%	-8.49%	-9.43%
W29	9.52%	0.37%	9.52%	-12.82%	-12.09%
W30	11.74%	-0.40%	10.53%	-10.93%	-12.96%
average	10.63%	0.21%	10.22%	-10.13%	-8.33%

Results indicate that FF is able to improve over RR on 10.63% in AMD and 8.07% in Intel. In turn, MU improves over RR on 0.22 (11%) in AMD and 0.13 (8%) in Intel. For all workloads studied, AMD hosts consumed more energy than Intel hosts. Moreover, if a performance model is added (with the characteristics studied in subsection 5.2), it is expected that the difference is even greater.

workload	FF	RD	MU	mU	mE
W1	11.93%	1.83%	11.93%	-11.93%	-11.01%
W2	9.62%	-2.88%	8.65%	-10.58%	-6.73%
W3	5.76%	-2.16%	6.47%	-7.19%	-7.91%
W4	6.63%	0.00%	7.23%	-6.63%	-6.02%
W5	4.05%	-2.03%	6.08%	-13.51%	-10.14%
W6	8.66%	0.79%	8.66%	-8.66%	-6.30%
W7	14.81%	6.67%	14.07%	-5.19%	-2.96%
W8	5.19%	-1.95%	5.19%	-9.74%	-5.19%
W9	4.97%	-1.24%	4.97%	-4.97%	-5.59%
W10	9.23%	2.31%	10.77%	-5.38%	-3.08%
W11	7.36%	-0.61%	7.98%	-11.04%	-5.52%
W12	10.13%	2.53%	10.76%	-6.33%	-5.06%
W13	10.92%	4.02%	12.07%	-9.20%	-6.32%
W14	4.14%	0.00%	4.83%	-6.21%	-6.90%
W15	5.85%	0.00%	6.43%	-5.26%	-4.68%
W16	-0.91%	-8.18%	0.00%	-18.18%	-17.27%
W17	6.67%	-1.48%	7.41%	-11.85%	-8.15%
W18	5.13%	-0.85%	5.13%	-5.98%	-5.98%
W19	8.50%	1.96%	7.19%	-7.19%	-5.88%
W20	10.98%	1.16%	10.98%	-5.78%	-6.94%
W21	13.29%	0.58%	13.29%	-12.72%	-10.40%
W22	15.66%	0.00%	15.06%	-12.05%	-6.63%
W23	5.36%	-0.89%	5.36%	-8.04%	-4.46%
W24	8.45%	3.52%	9.15%	-2.11%	-2.11%
W25	8.84%	1.36%	10.88%	-4.08%	-2.04%
W26	7.10%	-1.18%	8.28%	-8.28%	-3.55%
W27	10.39%	4.55%	10.39%	-7.79%	-3.25%
W28	5.81%	-0.65%	8.39%	-7.10%	-7.74%
W29	7.84%	0.49%	7.84%	-9.80%	-11.76%
W30	9.68%	0.00%	8.60%	-8.60%	-9.68%
average	8.07%	0.26%	8.47%	-8.38%	-6.64%

Table 6.14: GAPs over RR of the scheduling strategies on Intel host

The improvements (GAP) of each scheduling over RR on AMD host are graphically presented in Fig. 6.11.

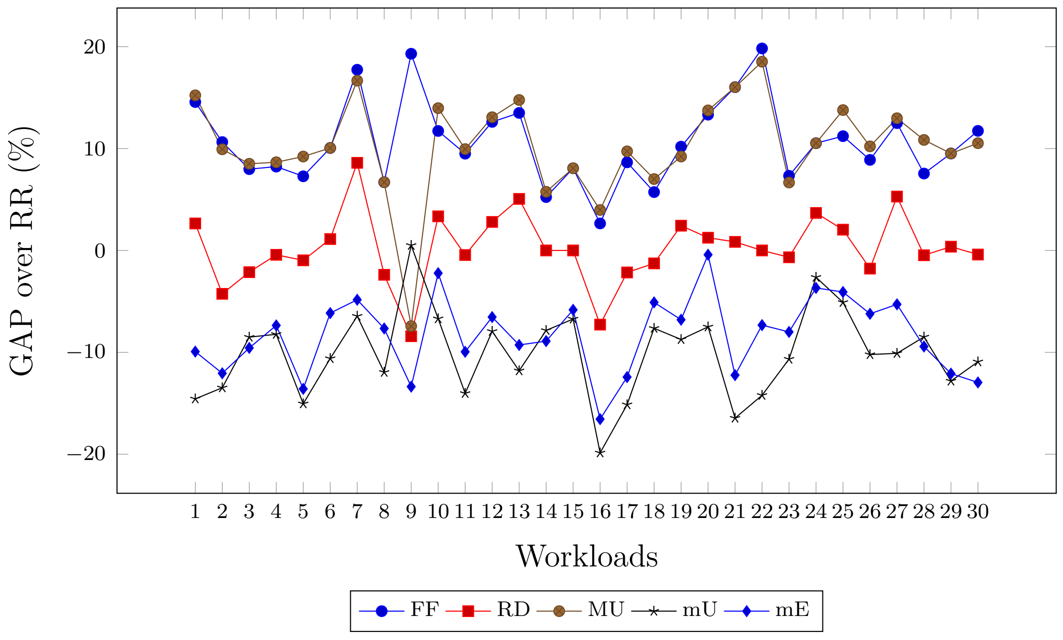


Figure 6.11: Percentage improvements in energy consumption over RR using AMD PC model

The improvements (GAP) of each scheduling over RR on Intel host are graphically presented in Fig. 6.12.

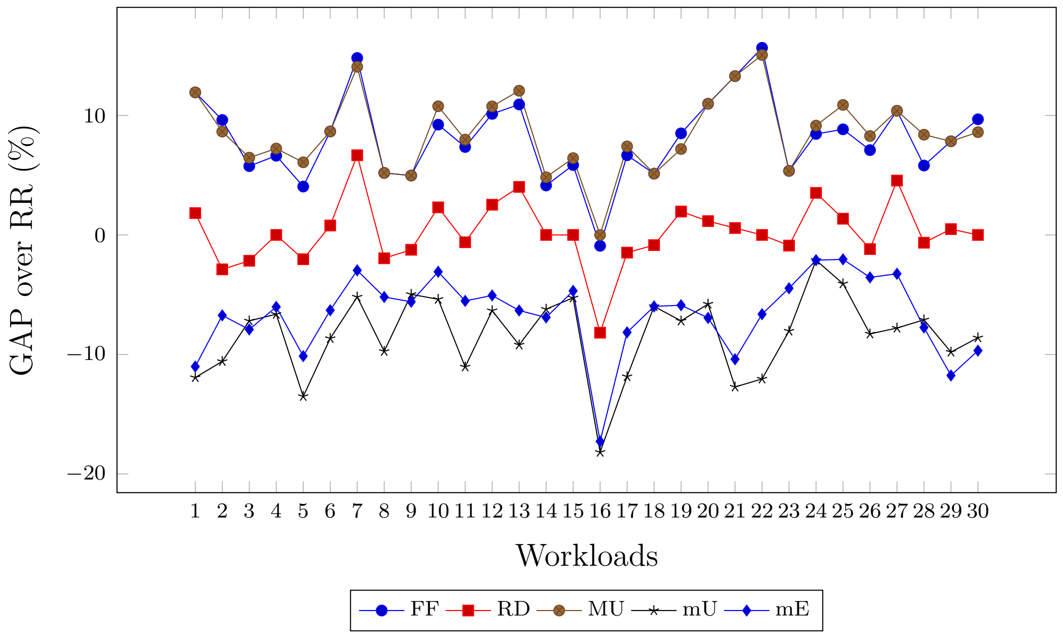


Figure 6.12: Percentage improvements in energy consumption over RR using Intel PC model

Overall, the reported results indicate that energy models built from power characterization and empirical data are important tools for the operation and management of HPC and data center infrastructures. Energy models allow evaluating energy-aware scheduling strategies with the main goal of reducing power consumption in large computing platforms.

Chapter 7

Conclusions and future work

This closing chapter presents the conclusions resulting from the study, and formulates the main lines of future work.

7.1 Conclusions

This thesis presented an empirical analysis of the energy consumption of synthetic benchmarks in high-end multi-core servers. In addition, power consumption models were constructed and used in simulations to evaluate several energy-aware scheduling strategies.

In the last decade, data centers have become a fundamental part of computers technology, since they are the backbone of the cloud, which is widely used by industrial and scientific applications worldwide. Furthermore, data centers are large consumers of energy, which implies a great challenge when it comes to consciously manage energy consumption, in order to reduce the environmental impact and the monetary costs. The first step towards achieving energy efficiency in data centers is to know the energy consumption of its different components, in particular, the physical hosts.

In this thesis, an exhaustive study of the inter-relationship among the power consumption of the main computing resources at different ULs was carried out over AMD and Intel architectures. All experiments were performed applying the reproducible research paradigm, which is not found often in this research area. All data and processing tools are publicly available for verification and extension by researchers. The experimental methodology consisted on executing synthetic benchmarks over high-end hosts connected to a PDU, considering different ULs and combinations of benchmarks with the goal of characterizing the power consumption of each computing resource (CPU, memory, and disk). The operations performed by the benchmarks include mathematical functions and read/write of main memory and disk.

The study was complemented with performance experiments. A total number of 144 experiments were performed: 96 experiments evaluating power consumption and 48 evaluating performance. For each experiment, 20 independent executions were performed. Results showed that in single executions, CPU utilization has a linear relation with power consumption. Memory utilization has significant impact on power consumption when compared to CPU usage, up to 157% more EC for AMD and 46% more EC for Intel. On the other hand, disk usage presented low EC variation for all ULs. Combined executions are able to reduce EC with regard to independent executions for CPU and disk combined execution, taking advantage of the optimum UL of computing resources. The results of two combined executions showed a remarkable energy reduction in the Intel host (32% of the EC of single executions). Efficiency analysis showed that different benchmarks performed more efficiently at different ULs: CPU at high ULs, memory at medium ULs, and disk at low ULs. The critic UL (100%) showed worse efficiency than high-medium UL (87.5%), except for disk.

Several statistical tools were applied to built a realistic power consumption model for multicores. The models built provide a good fit to the empirical data according to relevant metrics for statistical models, and they are useful tools to assess the capabilities of scheduling strategies regarding energy efficiency.

In order to evaluate energy-aware scheduling strategies, a total of 60 different simulations were performed considering linear energy models for AMD and Intel architectures and 30 realistic workloads based on traces of HPC Parallel Workloads Archive executed on real computing infrastructures. The simulation results showed that strategies which maximize the host utilization achieves better results, notably improving over traditional Business-as-Usual schedulers.

The research addressed by this thesis confirms that power consumption models built from empirical data are a key tool for operation and management of data center infrastructures. They allow evaluating different scheduling strategies via simulations to provide accurate methods that account for environmental and monetary savings.

Summarizing, the main contributions of the research reported in this thesis are:

- A cutting-edge review of related works regarding power characterization, modeling, energy-aware scheduling in cloud computing and supercomputing systems, and available cloud simulation tools.
- An empirical study of power consumption using benchmarks intensive on the three main computing resources that most contribute to power consumption utilization (CPU, memory, and disk), on two high-end multi-core servers (AMD and Intel architectures).
- An empirical study of performance degradation in multi-core servers with respect to the server load and the type of computing resource.
- Several energy models from experimental power data using supervised computational intelligence techniques.
- Energy evaluation through simulations for six scheduling strategies using realistic workloads.
- Reproducible/replicable research by using a Jupyter Notebook, showing in a clear and understandable manner the data processing from raw data.

In the context of this thesis, a collaboration with the research group at Centro de Investigación Científica y de Superior de Ensenada (CICESE) in Mexico was established. The main concepts of this investigation were discussed in the early stages of the investigation.

The research reported in this thesis resulted in two publications, which address some of the topics included in this manuscript. The first publication was a conference article, entilted *Power consumption characterization of synthetic benchmarks in multicores*, presented at *Latin American High Performance Computing Conference*, held in Buenos Aires, Argentina, in 2018 (Muraña et al., 2018). The conference article was written in colaboration with the researchers Sergio Nesmachnow and Santiago Iturriaga from Universidad de la República, Uruguay, and with the resercher Andrei Tcherniykh, from CICESE, Mexico. Figure 7.1 shows the first page of the conference article.

Power Consumption Characterization of Synthetic Benchmarks in Multicores

Jonathan Muraña[1](✉), Sergio Nesmachnow[1], Santiago Iturriaga[1], and Andrei Tchernykh[2]

[1] Universidad de la República, Montevideo, Uruguay
{jmurana,sergion,siturria}@fing.edu.uy
[2] CICESE Research Center, Ensenada, Baja California, Mexico
chernykh@cicese.mx

Abstract. This article presents an empirical evaluation of power consumption of synthetic benchmarks in multicore computing systems. The study aims at providing an insight of the main power consumption characteristics of different applications when executing over current high performance computing servers. Three types of applications are studied executing individually and simultaneously on the same server. Intel and AMD architectures are studied in an experimental setting for evaluating the overall power consumption of each application. The main results indicate the power consumption behavior has a strong dependency with the type of application. An additional performance analysis shows that the best load of the server regarding energy efficiency depends on the type of the applications, with efficiency decreasing in heavily loaded situations. These results allow characterizing applications according to power consumption, efficiency, and resource sharing, and provide useful information for resource management and scheduling policies.

Keywords: Green computing · Energy efficiency · Multicores Computing efficiency

1 Introduction

Nowadays, data centers are key infrastructures for executing industrial and scientific applications. Data centers have become highly popular for providing storage, computing power, middleware software, and others information technology (IT) utilities, available to researchers with ubiquitous access [3]. However, their energy efficiency has become a major concern in recent years, having a significant impact on monetary cost, environment, and guarantees for service-level agreements (SLA) [4].

The main sources of power consumption in data centers are the computational resources and the cooling system [13]. When focusing on power consumption due to resource utilization, several techniques for hardware and software optimization can be applied to improve energy efficiency. Software characterization techniques [1] are used to determine features that are useful to analyze

© Springer International Publishing AG 2018
E. Mocskos and S. Nesmachnow (Eds.): CARLA 2017, CCIS 796, pp. 21–37, 2018.
https://doi.org/10.1007/978-3-319-73353-1_2

Figure 7.1: Fisrt page of the conference article

The second publication was a journal article, entilted *Characterization, modeling and scheduling of power consumption of scientific computing applications in multicores*, published in *Cluster Computing* in 2019 (Muraña et al., 2019). The journal article was written in colaboration with Sergio Nesmachnow, Fermin Armenta (CICESE, Mexico), and Andrei Tcherniykh. Figure 7.2 shows the first page of the journal article.

Cluster Computing
https://doi.org/10.1007/s10586-018-2882-8

CrossMark

Characterization, modeling and scheduling of power consumption of scientific computing applications in multicores

Jonathan Muraña[1] · Sergio Nesmachnow[1] · Fermín Armenta[2] · Andrei Tchernykh[2]

Received: 2 April 2018 / Revised: 14 November 2018 / Accepted: 3 December 2018

Abstract
This article presents an empirical evaluation of power consumption for scientific computing applications in multicore systems. Three types of applications are studied, in single and combined executions on Intel and AMD servers, for evaluating the overall power consumption of each application. The main results indicate that power consumption behavior has a strong dependency with the type of application. Additional performance analysis shows that the best load of the server regarding energy efficiency depends on the type of the applications, with efficiency decreasing in heavily loaded situations. These results allow formulating a model to characterize applications according to power consumption, efficiency, and resource sharing, which provide useful information for resource management and scheduling policies. Several scheduling strategies are evaluated using the proposed energy model over realistic scientific computing workloads. Results confirm that strategies that maximize host utilization provide the best energy efficiency and performance results.

Keywords Green computing · Energy efficiency · Multicores · Energy model · Cloud simulator

1 Introduction

Data centers are key infrastructures for developing and executing industrial and scientific applications. In the last decade, data centers have become highly popular for providing storage, computing power, hosting, middleware software, and others information technology utilities, available to researchers with ubiquitous access [1]. However, energy efficiency of data centers has become one of the main concerns in recent years, having a significant impact on monetary cost, environment, and guarantees for service-level agreements (SLA) [2].

The main sources of power consumption in data centers are the computational resources and the cooling system [3]. Regarding power consumption of the computational resources, several techniques for hardware and software optimization can be applied to improve energy efficiency. For example, software characterization techniques [4], which are applied to determine features that are useful to analyze the software behavior, can be used to analyze and improve power consumption [5]. Estimation of power consumption is an important issue that can determine the quality, and even the feasibility, of both software products and big data center infrastructures.

Energy consumption models are useful synthetic tools for studying and analyzing diverse issues related to energy efficiency of computing infrastructures, e.g., by predicting the power consumption of its components for a given workload. Information resulting from the analysis is often used by decision-makers to take technical, monetary, or environmental decisions regarding the computing infrastructure. Energy models are based on the relation between power consumption and resource utilization and can be classified as hardware- or software-centric [2]. Models that follow complex approaches, i.e., by modeling the contribution of every hardware component and/or using machine

✉ Jonathan Muraña
jmurana@fing.edu.uy

Sergio Nesmachnow
sergion@fing.edu.uy

Fermín Armenta
armentac@cicese.edu.mx

Andrei Tchernykh
chernykh@cicese.edu.mx

1 Universidad de la República, Montevideo, Uruguay

2 CICESE Research Center, Ensenada, Mexico

Published online: 04 January 2019

Springer

Figure 7.2: Fisrt page of the journal article

7.2 Future work

The work presented in this thesis is the first step towards the construction of an energy-efficiency management of datacenters, guided by computational intelligence.

In order to extend the power and performance characterization, the main lines for future work are related to including different benchmarks and real scientific computing applications. Regarding benchmarks, network-bound and

GPU-based programs could be studied, mainly due to their relevance in modern datacenters and supercomputing infrastructures. The power characterization of non-synthetic benchmarks and programs from different scientific fields (e.g., molecular dynamics, quantum chemistry, computational fluid dynamics, climate modeling, etc.) will provide a useful insight for developers and scientist, which will certainly complement the analysis presented in this thesis. Considering other high-end hosts with different chip designs is also a possible line for future work, in order to extend the hardware components on the study. More complex energy models can also be considered, including features of combined executions, and enhancing the planning capabilities by considering resource utilization models, also derived from empirical data.

Extending the characterization of power consumption and building complex models can lead to designing powerful scheduling strategies. These strategies must consider energy and QoS together as an optimization objective, to meet the requirements of modern supercomputing facilities.

Bibliography

ACM. Association for Computing Machinery. Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-badging, 2018. Online; accessed December 21 2018.

A. Anghel, L. Vasilescu, G. Mariani, R. Jongerius, and G. Dittmann. An instrumentation approach for hardware-agnostic software characterization. *International Journal of Parallel Programming*, 44(5):924–948, 2016.

F. Armenta-Cano, A. Tchernykh, J. Cortes-Mendoza, R. Yahyapour, A. Drozdov, P. Bouvry, D. Kliazovich, A. Avetisyan, and S. Nesmachnow. Min_c: Heterogeneous concentration policy for energy-aware scheduling of jobs with resource contention. *Programming and Computer Software*, 43(3):204–215, 2017.

S. Bak, M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek, and J. Waglarz. GSSIM–a tool for distributed computing experiments. *Scientific Programming*, 19(4):231–251, 2011.

L. Barroso, J. Clidaras, and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

T. Bawden. Global warming: Data centres to consume three times as much energy in next decade. https://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html, 2016. Online; accessed January 14 2019.

C. Begley. Six red flags for suspect work. *Nature*, 497(7450):433–434, 2013.

A. Beloglazov, R. Buyya, Y. Choon Lee, and A. Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82, 2010.

A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28:755 – 768, 2012.

C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

C. Brandolese, S. Corbetta, and W. Fornaciari. Software energy estimation based on statistical characterization of intermediate compilation code. In *International Symposium on Low Power Electronics and Design*, pages 333–338, 2011.

R. Buyya, C. Vecchiola, and S. Selvi. *Mastering Cloud Computing: Foundations and Applications Programming.* Morgan Kaufmann, San Francisco, CA, USA, 2013.

R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

F. Chen, J. Grundy, Y. Yang, J. Schneider, and Q. He. Experimental analysis of task-based energy consumption in cloud computing systems. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 295–306, 2013.

H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu. Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment. *Journal of Systems and Software*, 99:20–35, 2015.

Cisco Systems. Cisco global cloud index: Forecast and methodology, 2015-2020. White paper. *Cisco Public, San Jose*, 2016.

D. Cox and N. Reid. *The theory of the design of experiments.* Chapman and Hall/CRC, 2000.

CPUBOSS. Intel Xeon E52643v3 vs AMD Opteron 6172 comparison. http://cpuboss.com/cpus/Intel-Xeon-E5-2643-v3-vs-AMD-Opteron-6172, 2014. Online; accessed March 29 2018.

R. Danilak. Why energy is a big and rapidly growing problem for data centers. https://www.forbes.com/sites/forbestechcouncil/2017/12/15/why-energy-is-a-big-and-rapidly-growing-problem-for-data-centers, 2017. Online; accessed January 14 2019.

M. Dayarathna, Y. Wen, and R. Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794, 2016.

R. Dick and Z. Mao. Cache contention and application performance prediction for multi-core systems. In *IEEE International Symposium on Performance Analysis of Systems Software*, pages 76–86, 2010.

J. Dongarra. The linpack benchmark: An explanation. In *Proceedings of the 1st International Conference on Supercomputing*, pages 456–474, London, UK, 1988. Springer-Verlag.

K. Du Bois, T. Schaeps, S. Polfliet, F. Ryckbosch, and L. Eeckhout. Sweep: Evaluating computer system energy efficiency using synthetic workloads. In *6th International Conference on High Performance and Embedded Architectures and Compilers*, pages 159–166, 2011.

H. El-Rewini, T. Lewis, and H. Ali. *Task Scheduling in Parallel and Distributed Systems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

D. Feitelson, D. Tsafrir, and D. Krakov. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing*, 74(10): 2967–2982, 2014.

X. Feng, R. Ge, and K. Cameron. Power and energy profiling of scientific applications on distributed systems. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 34–44, 2005.

F. Fernandes, D. Beserra, E. Moreno, B. Schulze, and R. Coelho. A virtual machine scheduler based on CPU and I/O-bound features for energy-aware in high performance computing clouds. *Computers Electrical Engineering*, 56:854 – 870, 2016.

S. Gade, H. Mondal, and S. Deb. A hardware and thermal analysis of DVFS in a multi-core system with hybrid WNoC architecture. In *28th International Conference on VLSI Design*, pages 117–122, 2015.

L. Galaviz-Alejos, F. Armenta-Cano, A. Tchernykh, G. Radchenko, A. Drozdov, O. Sergiyenko, and R. Yahyapour. Bi-objective heterogeneous consolidation in cloud computing. In *High Performance Computing*, pages 384–398, Cham, 2018.

J. Gao. Machine learning applications for data center optimization. https://ai.google/research/pubs/pub42542, 2014. Online; accessed January 14 2019.

Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu. Service level agreement based energy-efficient resource management in cloud data centers. *Computers & Electrical Engineering*, 40(5):1621–1633, 2014.

X. Gonze, B. Amadon, P. Anglade, J. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, P. Ghosez, M. Giantomassi, S. Goedecker, D. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M. Verstraete, G. Zerah, and J. Zwanziger. ABINIT: First-principles approach to material and nanosystem properties. *Computer Physics Communications*, 180(12):2582 – 2615, 2009.

S. Goodman, D. Fanelli, and J. Ioannidis. What does research reproducibility mean? *Science Translational Medicine*, 8:341ps12–341ps12, 2016.

Google LLC. Efficiency: How we do it. https://www.google.com/about/datacenters/efficiency/internal, 2019a. Online; accessed January 14 2019.

Google LLC. Google Colab. https://colab.research.google.com/, 2019b. Online; accessed April 2019.

R. Grant, J. Laros, M. Levenhagen, S. Olivier, K. Pedretti, L. Ward, and A. Younge. Evaluating energy and power profiling techniques for HPC workloads. In *Eighth International Green and Sustainable Computing Conference*, pages 1–8, 2017.

V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke. Smart grid technologies: Communication technologies and standards. *IEEE Transactions on Industrial Informatics*, 7(4):529–539, 2011.

Z. Guo, S. Hui, Y. Xu, and H. Chao. Dynamic flow scheduling for power-efficient data center networks. In *IEEE/ACM 24th International Symposium on Quality of Service*, pages 1–10, 2016.

M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, New York, USA, 1st edition, 2013.

S. Hernández, J. Fabra, P. Álvarez, and J. Ezpeleta. Simulation and realistic workloads to support the meta-scheduling of scientific workflows. In *Simulation and Modeling Methodologies, Technologies and Applications*, pages 155–167, Cham, 2014.

IBM. IBM Watson Studio. https://www.ibm.com/cloud/watson-studio/, 2019. Online; accessed April 2019.

C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: methodology and empirical data. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–104, 2003.

S. Iturriaga, S. García, and S. Nesmachnow. An empirical study of the robustness of energy-aware schedulers for high performance computing systems under uncertainty. In *High Performance Computing*, pages 143–157, Cham, 2014.

D. Jackson, Q. Snell, and M. Clement. Core algorithms of the Maui scheduler. In *Job Scheduling Strategies for Parallel Processing*, pages 87–102, 2001.

C. Jiang, C. Wang, X. Liu, and Y. Zhao. A survey of job scheduling in grids. In G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, editors, *Advances in Data and Web Management*, pages 419–427, 2007.

Jupyter Community. Project jupyter. http://jupyter.org. Online; accessed 01 March 2018.

T. Kaur and I. Chana. Energy aware scheduling of deadline-constrained tasks in cloud computing. *Cluster Computing*, 19(2):679–698, 2016.

D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. In *IEEE Global Telecommunications Conference*, pages 1–5, 2010.

T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks: a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, 2016.

A. Kopytov. Sysbench repository. https://github.com/akopytov/sysbench, 2017. Online; accessed June 01 2017.

K. Kurowski, A. Oleksiak, W. Piątek, T. Piontek, A. Przybyszewski, and J. Węglarz. DCworms–a tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, 39: 135–151, 2013.

A. Langer, E. Totoni, U. Palekar, and L. Kalé. Energy-efficient computing for HPC workloads on heterogeneous manycore chips. In *Proceedings of the 6th International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 11–19, 2015.

J. Laros, P. Pokorny, and D. DeBonis. PowerInsight - a commodity power measurement capability. In *International Green Computing Conference Proceedings*, pages 1–6, 2013.

E. Le Sueur and G. Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the international conference on power aware computing and systems*, pages 1–8, 2010.

J. Leung, L. Kelly, and J. Anderson. *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

R. Malhotra and P. Jain. Study and comparison of various cloud simulators available in the cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(9):347–350, 2013.

K. Malladi, F. Nothaft, K. Periyathambi, B. Lee, C. Kozyrakis, and M. Horowitz. Towards energy-proportional datacenter memory with mobile DRAM. In *39th Annual International Symposium on Computer Architecture*, pages 37–48, 2012.

S. Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, London, UK, 2011.

W. McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

K. Millman and M. Aivazis. Python for scientists and engineers. *Computing in Science Engineering*, 13(2):9–12, 2011.

A. Mishra and A. Tripathi. Energy efficient voltage scheduling for multi-core processors with software controlled dynamic voltage scaling. *Applied Mathematical Modelling*, 38(14):3456–3466, 2014.

B. Moreno, A. López, and M. García-Álvarez. The electricity prices in the european union. the role of renewable energies and regulatory electric market reforms. *Energy*, 48(1):307 – 313, 2012.

J. Muraña, S. Iturriaga, and S. Nesmachnow. A multiobjective evolutionary algorithm for QoS-aware planning in heterogeneous computing systems. In *XL Latin American Computing Conference*, pages 1–12, 2014.

J. Muraña, S. Nesmachnow, S. Iturriaga, and A. Tchernykh. Power consumption characterization of synthetic benchmarks in multicores. In *High Performance Computing*, pages 21–37, Cham, 2018.

J. Muraña, S. Nesmachnow, F. Armenta, and A. Tchernykh. Characterization, modeling and scheduling of power consumption of scientific computing applications in multicores. *Cluster Computing*, 2019. URL https://link.springer.com/article/10.1007/s10586-018-2882-8. Online first.

S. Nesmachnow. Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. *Revista de la Asociación de Ingenieros del Uruguay*, 61(1):12–15, 2010. Text in Spanish.

S. Nesmachnow and J. Muraña. Multiobjective scheduling with service levels in heterogeneous computing systems. In *VIII ALIO/EURO Workshop on Applied Combinatorial Optimization*, 2014.

S. Nesmachnow, B. Dorronsoro, J. Pecero, and P. Bouvry. Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, 11(4):653–680, 2013.

S. Nesmachnow, C. Perfumo, and Í. Goiri. Multiobjective energy-aware datacenter planning accounting for power consumption profiles. In *High Performance Computing*, volume 485, pages 128–142. 2014.

S. Nesmachnow, C. Perfumo, and I. Goiri. Holistic multiobjective planning of datacenters powered by renewable energy. *Cluster Computing*, 18(4):1379–1397, 2015.

P. Nielsen. cpuburn. https://github.com/patrickmn/cpuburn, 2012. Online; accessed April 2019.

A. Núñez, J. Vázquez-Poletti, A. Caminero, G. Castañé, J. Carretero, and I. Llorente. ICanCloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

T. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing, USA, 2006.

T. Oliphant. Python for scientific computing. *Computing in Science Engineering*, 9(3):10–20, 2007.

P. Patil, R. Peng, and J. Leek. A statistical definition for reproducibility and replicability. *bioRxiv*, 2016. URL https://www.biorxiv.org/content/early/2016/07/29/066803.full.pdf. Online; accessed December 21 2018.

A. Peckov. *A Machine Learning Approach to Polynomial Regression*. PhD thesis, Jozef Stefan International Postgraduate School, Ljubljana, 2012.

J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, and K. Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

M. Pinedo. *Scheduling: theory, algorithms, and systems.* Springer, 2016.

H. Plesser. Reproducibility vs. replicability: A brief history of a confused terminology. *Frontiers in Neuroinformatics*, 11:76, 2018.

R Core Team. *R: A Language and Environment for Statistical Computing*, 2018. URL https://www.R-project.org. Online; accessed December 21 2018.

M. Rashti, G. Sabin, D. Vansickle, and B. Norris. WattProf: A flexible platform for fine-grained HPC power profiling. In *IEEE International Conference on Cluster Computing*, pages 698–705, 2015.

A. Repko. *Interdisciplinary research : process and theory.* SAGE Publishing, Los Angeles, USA, 2008.

H. Rong, H. Zhang, S. Xiao, C. Li, and C. Hu. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674 – 691, 2016.

S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner. United states data center energy usage report. Technical report, Lawrence Berkeley National Laboratory, 06 2016.

W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts.* Wiley Publishing, 9th edition, 2012.

J. Sousa and L. Wolsey. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical programming*, 54(1-3):353–367, 1992.

S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Conference on Power Aware Computing and Systems*, pages 1–5, 2008.

A. Tchernykh, L. Lozano, P. Bouvry, J. Pecero, U. Schwiegelshohn, and S. Nesmachnow. Energy-aware online scheduling: Ensuring quality of service for IaaS clouds. In *International Conference on High Performance Computing Simulation*, pages 911–918, 2014.

H. Theil. *Economic forecasts and policy.* Amsterdam North-Holland Publishing Company, 1961.

E. Totoni, N. Jain, and L. Kalé. Toward runtime power management of exascale networks by on/off control of links. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 915–922, 2013.

J. Treibig, G. Hager, and G. Wellein. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *39th International Conference on Parallel Processing Workshops*, pages 207–216, 2010.

Z. Zhan, X. Liu, Y. Gong, J. Zhang, H. Chung, and Y. Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys*, 47(4):63:1–63:33, 2015.

H. Zhang and H. Hoffmann. Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. *ACM SIGARCH Computer Architecture News*, 44(2):545–559, 2016.

Y. Zhang and N. Ansari. HERO: Hierarchical energy optimization for data center networks. *IEEE Systems Journal*, 9(2):406–415, 2015.

S. Zhuravlev, S. Blagodurov, and A. Fedorova. Addressing shared resource contention in multicore processors via scheduling. In *ACM Sigplan Notices*, volume 45, pages 129–142, 2010.